VAX-11/RSX-11M
Programmer's Reference Manual

Order No. AA-D020C-TE

May 1982

This document describes VAX/VMS support of the RSX-11M Executive directives. It contains the information needed by an RSX-11M programmer responsible for making RSX-11M Version 3.2 task images run under VAX/VMS.

REVISION/UPDATE INFORMATION: This revised document supersedes the VAX-11/RSX-11M Programmer's Reference Manual (Order No. AA-D020B-TE).

SOFTWARE VERSION: VAX/VMS V3.0

digital equipment corporation · maynard, massachusetts

First Printing, August 1978
Second Printing, March 1980
Revised, May 1982

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	RSX
DEC/CMS	EduSystem	UNIBUS
DECnet	IAS	VAX
DECsystem-10	MASSBUS	VMS
DECSYSTEM-20	PDP	VT
DECUS	PDT	digital
DECwriter	RSTS	

ZK2138

HOW TO ORDER ADDITIONAL DOCUMENTATION

In Continental USA and Puerto Rico call 800-258-1710

In New Hampshire, Alaska, and Hawaii call 603-884-6660

In Canada call 613-234-7726 (Ottawa-Hull)
800-267-6146 (all other Canadian)

DIRECT MAIL ORDERS (USA & PUERTO RICO)*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire 03061

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575)

DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.
940 Belfast Road
Ottawa, Ontario K1G 4C2
Attn: A&SG Business Manager

DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation
A&SG Business Manager
c/o Digital's local subsidiary or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

CONTENTS

		Page
PREFACE		vii
SUMMARY OF TECHNICAL CHANGES		ix
CHAPTER 1	INTRODUCTION	
1.1	VAX-11 COMPATIBILITY WITH PDP-11 PROCESSORS	1-2
1.2	VAX/VMS COMPATIBILITY WITH RSX-11M VERSION 3.2	1-2
1.3	RSX-11M DIRECTIVE REQUESTS	1-3
1.4	RSX-11M MEMORY HANDLING OPTIONS	1-4
1.5	EMULATION OF FLOATING-POINT INSTRUCTIONS	1-4
1.6	VAX/VMS SYSTEM CONCEPTS	1-5
1.7	DISTINCTION BETWEEN PROGRAMMING AND SYSTEM ENVIRONMENTS	1-5
CHAPTER 2	THE VAX/VMS SYSTEM ENVIRONMENT	
2.1	PRIVILEGES	2-1
2.2	UIC-BASED PROTECTION	2-1
2.3	RESOURCE USAGE LIMITS	2-2
2.4	PROCESS NAMES	2-3
2.5	EVENT FLAG CLUSTERS	2-4
2.6	SYSTEM STATUS CODES	2-5
2.7	MEMORY MANAGEMENT	2-6
2.7.1	Balance Sets and Working Sets	2-6
2.7.2	Functions of the Swapper	2-6
2.7.3	Functions of the Pager	2-7
2.8	SYSTEM EVENTS	2-7
2.9	SYSTEM CLOCK	2-8
2.10	SOFTWARE PRIORITIES	2-8
2.11	GLOBAL SECTIONS	2-8
2.12	HIBERNATION	2-10
2.13	IMAGE TERMINATION	2-10
2.13.1	Normal Termination	2-10
2.13.2	Abnormal Termination	2-11
2.14	PARSING OF FILE SPECIFICATIONS	2-13
2.15	VAX/VMS I/O SYSTEM	2-13
CHAPTER 3	VAX/VMS I/O SYSTEM	
3.1	VAX-11 RECORD MANAGEMENT SERVICES (VAX-11 RMS)	3-1
3.2	VAX/VMS I/O SYSTEM SERVICES	3-2
3.2.1	Assign I/O Channel System Service	3-3
3.2.2	Queue I/O Request System Service	3-3
3.2.3	Create Mailbox and Assign I/O Channel System Service	3-3
3.2.4	Additional I/O System Services	3-4
3.3	I/O DRIVERS AND ACPS	3-4
3.4	RSX-11M IMAGE INTERFACE TO THE VAX/VMS I/O SYSTEM	3-5

		Page
3.5	DEVICE ASSIGNMENT	3-5
3.6	DEVICE MAPPING	3-7
3.6.1	Mapping Pseudodevice Names	3-8
3.6.2	The LB Pseudodevice Name and Concealed Devices	3-8
3.7	HANDLING OF QUEUE I/O FUNCTION CODES	3-9
3.8	MAILBOXES	3-10
3.8.1	Mailboxes for Send/Receive Directives	3-10
3.8.2	I/O to Mailboxes	3-10
3.9	ACP FUNCTIONS	3-11
3.10	SPOOLED DEVICES	3-11
3.10.1	FCS Spooling	3-12

CHAPTER 4 DIRECTIVE DESCRIPTIONS

4.1	VAX/VMS HANDLING OF DIRECTIVES	4-1
4.2	SYSTEM DIRECTIVE DESCRIPTIONS	4-6
4.2.1	ABRT$ - ABORT TASK	4-8
4.2.2	ALTP$ - ALTER PRIORITY	4-9
4.2.3	ALUN$ - ASSIGN LUN	4-10
4.2.4	ASTX$S - AST SERVICE EXIT	4-11
4.2.5	CLEF$ - CLEAR EVENT FLAG	4-12
4.2.6	CMKT$ - CANCEL MARK TIME REQUESTS	4-13
4.2.7	CRGF$ - CREATE GROUP GLOBAL EVENT FLAGS	4-14
4.2.8	CSRQ$ - CANCEL TIME BASED INITIATION REQUESTS	4-15
4.2.9	DECL$S - DECLARE SIGNIFICANT EVENT	4-16
4.2.10	DSAR$S (or IHAR$S) - DISABLE (or INHIBIT) AST RECOGNITION	4-17
4.2.11	DSCP$S - DISABLE CHECKPOINTING	4-18
4.2.12	ELGF$ - ELIMINATE GROUP GLOBAL EVENT FLAGS	4-19
4.2.13	ENAR$S - ENABLE AST RECOGNITION	4-20
4.2.14	ENCP$S - ENABLE CHECKPOINTING	4-21
4.2.15	EXIF$ - EXIT IF	4-22
4.2.16	EXIT$S - TASK EXIT	4-23
4.2.17	EXST$ - EXIT WITH STATUS	4-24
4.2.18	EXTK$ - EXTEND TASK	4-25
4.2.19	GLUN$ - GET LUN INFORMATION	4-26
4.2.20	GMCR$ - GET MCR COMMAND LINE	4-28
4.2.21	GPRT$ - GET PARTITION PARAMETERS	4-30
4.2.22	GTIM$ - GET TIME PARAMETERS	4-31
4.2.23	GTSK$ - GET TASK PARAMETERS	4-32
4.2.24	MRKT$ - MARK TIME	4-33
4.2.25	QIO$ - QUEUE I/O REQUEST	4-35
4.2.26	QIOW$ - QUEUE I/O REQUEST AND WAIT	4-37
4.2.27	RCST$ - RECEIVE DATA OR STOP	4-38
4.2.28	RCVD$ - RECEIVE DATA	4-40
4.2.29	RCVX$ - RECEIVE DATA OR EXIT	4-41
4.2.30	RDAF$ - READ ALL EVENT FLAGS	4-42
4.2.31	RDXF$ - READ EXTENDED EVENT FLAGS	4-43
4.2.32	RQST$ - REQUEST TASK	4-44
4.2.33	RSUM$ - RESUME TASK	4-45
4.2.34	RUN$ - RUN TASK	4-46
4.2.35	SDAT$ - SEND DATA	4-48
4.2.36	SETF$ - SET EVENT FLAG	4-50
4.2.37	SFPA$ - SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST	4-51
4.2.38	SPND$S - SUSPEND	4-52
4.2.39	SPRA$ - SPECIFY POWER RECOVERY AST	4-53
4.2.40	SPWN$ - SPAWN	4-54
4.2.41	SRDA$ - SPECIFY RECEIVE DATA AST	4-55
4.2.42	STLO$ - STOP FOR LOGICAL OR OF EVENT FLAGS	4-57
4.2.43	STOP$S - STOP	4-58
4.2.44	STSE$ - STOP FOR SINGLE EVENT FLAG	4-59
4.2.45	SVDB$ - SPECIFY SST VECTOR TABLE FOR DEBUGGING AID	4-60

Page

4.2.46 SVTK$ - SPECIFY SST VECTOR TABLE FOR TASK . . . 4-61
4.2.47 USTP$ - UNSTOP TASK 4-62
4.2.48 WSIG$S - WAIT FOR SIGNIFICANT EVENT 4-64
4.2.49 WTLO$ - WAIT FOR LOGICAL OR OF EVENT FLAGS . . 4-65
4.2.50 WTSE$ - WAIT FOR SINGLE EVENT FLAG 4-66

CHAPTER 5 I/O DRIVERS

5.1 SUPPORTED DEVICES 5-2
5.2 GET LUN INFORMATION DIRECTIVE 5-2
5.3 STANDARD I/O FUNCTIONS 5-2
5.3.1 Attach and Detach I/O Device (IO.ATT and IO.DET) 5-2
5.3.2 Cancel I/O Requests (IO.KIL) 5-3
5.4 I/O STATUS BLOCK AND STATUS RETURNS 5-3
5.5 DISK DRIVER 5-8
5.6 MAGNETIC TAPE DRIVER 5-9
5.7 LINE PRINTER DRIVER 5-10
5.8 TERMINAL DRIVER 5-11
5.8.1 IO.ATT Function 5-15
5.8.1.1 IO.ATT!TF.AST and IO.ATA Functions 5-15
5.8.1.2 IO.ATT!TF.ESQ Function 5-16
5.8.2 IO.DET Function 5-16
5.8.3 IO.KIL function 5-16
5.8.4 IO.RLB, IO.RAL, IO.RNE, IO.RST, and IO.RTT Functions 5-16
5.8.4.1 IO.RLB!TF.RAL and IO.RAL 5-16
5.8.4.2 IO.RLB!TF.RNE and IO.RNE Functions 5-17
5.8.4.3 IO.RLB!TF.RST and IO.RST Functions 5-17
5.8.4.4 IO.RLB!TF.RTT and IO.RTT Functions 5-17
5.8.5 IO.RPR Function 5-18
5.8.5.1 IO.RPR!TF.XOF Function 5-18
5.8.6 IO.RVB Function 5-18
5.8.7 IO.RPB Function 5-18
5.8.8 IO.WLB, IO.CCO, and IO.WBT Functions 5-18
5.8.8.1 IO.WLB!TF.CCO and IO.CCO Functions 5-19
5.8.8.2 IO.WLB!WBT and IO.WBT Functions 5-19
5.8.9 IO.WVB Function 5-19
5.8.9.1 IO.WLB!TF.WAL, IO.WAL, and IO.CCO!TF.WAL Functions 5-19
5.8.10 IO.WPB Function 5-19
5.8.11 IO.GTS Function 5-19
5.8.12 SF.GMC Function 5-21
5.8.13 SF.SMC Function 5-21
5.8.14 Terminal Read Status Returns 5-22
5.9 CARD READER DRIVER 5-23
5.10 NULL DEVICE 5-23
5.11 DISK AND MAGNETIC TAPE ACPS 5-23
5.11.1 General Correspondence of Parameters 5-26
5.11.2 IO.CRE Function 5-26
5.11.3 IO.DEL with EX.ENA=0 5-27
5.11.4 IO.DEL with EX.ENA=1 5-27
5.11.5 IO.ACR Function 5-27
5.11.6 IO.ACW and IO.ACE Functions 5-28
5.11.7 IO.DAC Function 5-28
5.11.8 IO.EXT Function 5-28
5.11.9 IO.WAT Function 5-29
5.11.10 IO.RAT Function 5-29
5.11.11 IO.FNA Function 5-29
5.11.12 IO.RNA Function 5-30
5.11.13 IO.ENA Function 5-30
5.11.14 IO.APC Function 5-31

CONTENTS

Page

APPENDIX A VAX-11 COMPATIBILITY MODE INSTRUCTION SET

APPENDIX B PARSE DIRECTIVE

B.1 NORMAL MODE PARSING B-1
B.2 DEVICE-ONLY PARSING B-2
B.3 DEFAULT FILENAME BLOCK PARSING B-2
B.4 RMS-11 PARSING B-2
B.5 DIRECTIVE CALL AND DPB FORMATS B-2

INDEX

FIGURES

FIGURE 1-1 Process Virtual Address Space 1-6
2-1 VAX/VMS UIC Format 2-1
3-1 Components of VAX/VMS I/O System 3-2
3-2 RSX-11M Image Interface to VAX/VMS I/O System . . 3-6
3-3 Use of Mailboxes for Send/Receive Directives . . 3-11
5-1 Format of RSX-11 I/O Status Block under VAX/VMS . 5-3
5-2 File Identification Block Format 5-25

TABLES

TABLE 2-1 Reasons for RSX-11M Image Termination 2-12
3-1 Device Name Mapping 3-8
4-1 VAX/VMS Handling of Directives 4-1
5-1 I/O Status Return Codes 5-4
5-2 Disk Function Code Correspondence 5-8
5-3 Disk Parameter Correspondence 5-9
5-4 Magnetic Tape Function Code Correspondence 5-9
5-5 Magnetic Tape Parameter Correspondence 5-10
5-6 Line Printer Function Code Correspondence . . . 5-10
5-7 Line Printer Parameter Correspondence 5-11
5-8 Terminal Function Code Correspondence 5-12
5-9 Terminal Parameter Correspondence 5-13
5-10 Subfunction Bit Correspondence 5-14
5-11 Information Returned by Get Terminal Support (IO.GTS) 5-20
5-12 Terminal Characteristics for SF.GMC and SF.SMC Requests 5-21
5-13 Card Reader Function Code Correspondence 5-23
5-14 ACP Parameter Correspondence 5-26
A-1 VAX-11 Compatibility Mode Instruction Set A-1

PREFACE

MANUAL OBJECTIVES

The VAX-11/RSX-11M Programmer's Reference Manual describes how VAX/VMS supports RSX-11M directives. This document bridges the gap between the RSX-11M/M-PLUS Executive Reference Manual and the VAX/VMS System Services Reference Manual, and between the RSX-11M/M-PLUS I/O Drivers Reference Manual and the VAX/VMS I/O User's Guide.

INTENDED AUDIENCE

This manual contains information needed by an RSX-11M programmer who is responsible for making RSX-11M task images run under VAX/VMS.

This document has two prerequisites:

- Understanding of the RSX-11M operating system and executive directives
- Understanding of the material presented in the VAX-11/RSX-11M User's Guide

STRUCTURE OF THIS DOCUMENT

Information in this document is organized as follows:

- Chapter 1 contains a general definition of VAX-11 compatibility mode and VAX/VMS Version 3.0 support of RSX-11M task images. It also contains a general description of basic VAX/VMS concepts, such as process and image, and their relationship to an RSX-11M task.
- Chapters 2 and 3 discuss certain VAX/VMS system components and explain the implications of their use for RSX-11M task images. Chapter 3 focuses on the use of the VAX/VMS I/O system for RSX-11M image I/O.
- Chapter 4 describes each RSX-11M directive as it is supported under VAX/VMS.
- Chapter 5 discusses QUEUE I/O REQUEST directive function codes and function-dependent parameters for devices supported by VAX/VMS.
- Appendix A contains the VAX-11 compatibility mode instruction set. Appendix B describes the VAX-11 RMS parse directive.

ASSOCIATED DOCUMENTS

The following documents may also be useful.

- VAX-11 Information Directory and Index
- VAX/VMS Primer
- VAX/VMS System Services Reference Manual
- VAX/VMS I/O User's Guide
- VAX/VMS Summary Description and Glossary
- VAX/VMS System Management and Operations Guide
- VAX-11 Record Management Services Reference Manual
- RSX-11M document set, preferably for RSX-11M Version 3.2

CONVENTIONS USED IN THIS DOCUMENT

This manual uses the same conventions as the RSX-11M/M-PLUS Executive Reference Manual. For example, brackets ([]) indicate optional parameters. In addition, the directive descriptions in Chapter 4 use shading to highlight differences in VAX/VMS support of the directives.

RELEASE NUMBERS AND OPERATING SYSTEMS

VAX/VMS compatibility mode allows RSX-11M programs to run on a VAX-11 system with a minimum amount of difficulty. Compatibility mode is a part of the VAX-11 operating system, VAX/VMS Version 3.0.

Compatibility mode generally supports the functions provided by RSX-11M Version 3.2. After RSX-11M Version 3.2, RSX-11M-PLUS Version 2.0 and RSX-11M Version 4.0 were introduced, providing new features not available in RSX-11M Version 3.2. Like RSX-11M Version 3.2, compatibility mode does not support the new features added to RSX-11M-PLUS Version 2.0 or RSX-11M Version 4.0.

Except for the new features added since RSX-11M Version 3.2 and certain features that cannot be supported because of inherent differences between the PDP-11 and VAX-11 systems, you can use compatibility mode to provide an environment that resembles RSX-11M on a VAX-11.

SUMMARY OF TECHNICAL CHANGES

Changes to this document include the following:

- Compatibility mode requirements changes (see Chapter 1)
- VAX/VMS changes (see Chapters 2 and 3)
- Disk driver and terminal driver changes (see Chapter 5)

For other VAX/VMS compatibility mode components, RSX-11M features added since RSX-11M Version 3.2 are generally not supported. For example, the RSX-11M directives described in Chapter 4 have not been changed to be compatible with versions of RSX-11M released since Version 3.2.

The following changes have occurred to the VAX/VMS equivalent of the RSX-11M disk driver:

- The Write Physical Block with Deleted Data Mark function, IO.WDD, is now supported for RX01 and RX02 diskettes. Your process must have LOG_IO privilege to perform this function.
- When you read data written using the Write Physical Block with Deleted Data Mark function using the Read Physical Block function, the success status of the I/O operation is returned as IS.RDD instead of IS.SUC.

The following changes have occurred to the VAX/VMS equivalent of the RSX-11M terminal driver:

- The SF.SMC function now supports the TC.TTP (set terminal type) characteristic, for the following terminals: LA36, LA120, LA34, LA38, VT05, VT52, VT55, VT100, VT101, VT102, VT105, VT125, VT131, and VT132. Other terminals are treated as "unknown terminals."
- The SF.SMC function now supports the TC.BIN (binary input mode) characteristic, where no characters are interpreted as control characters.

Additional information has been added to Chapter 5 about the general differences between the RSX-11M and compatibility mode terminal drivers, in regard to the <CR> and <LF> characters sent in response to terminal read and write requests.

CHAPTER 1

INTRODUCTION

Compatibility mode is a processor state that allows PDP-11 programs to execute under the VAX-11 processor. For compatibility mode execution to occur, the needs of the program must be satisfied on two levels:

- At the hardware instruction set level
- At the level of program interface to the operating system

At the hardware level, the VAX-11 processor includes an instruction set that is a compatible subset of the PDP-11 instruction set. This compatibility mode instruction set provides a general basis that potentially allows any PDP-11 user mode program to execute using the VAX-11 hardware under the VAX-11 operating system, VAX/VMS.

In addition, VAX/VMS supplements the subset of PDP-11 instructions that can be used in compatibility mode through software emulation of the FPP floating-point instructions; FIS floating-point instructions and EAE (Extended Arithmetic Element) instructions are not emulated.

Because of the two instruction sets, the VAX-11 processor has two basic modes of operation: native and compatibility. The processor operates in native mode to execute native mode instructions; it operates in compatibility mode to execute PDP-11 instructions. Software controls the processor mode. Thus, when a compatibility program has been prepared for execution, VAX/VMS places the processor in compatibility mode just before passing control to the program. VAX/VMS accomplishes this in a manner that is transparent to the user.

When an RSX-11M task image executing in VAX-11 compatibility mode attempts to interface with the operating system, a hardware-generated trap occurs. Hardware-generated traps are the mechanism by which the processor notifies VAX/VMS that emulation of the RSX-11M operating system's environment is required. The occurrence of a compatibility mode trap automatically places the processor in native mode.

Executing in native mode, VAX/VMS duplicates the RSX-11M task/system interface. VAX/VMS returns to the task in compatibility mode to allow the task to continue execution.

For example, RSX-11M tasks use EMT 377 instructions to interface with the operating system. An attempt to execute an EMT 377 instruction on VAX-11 hardware causes a trap to VAX/VMS. VAX/VMS then emulates the requested service in native mode, places the processor in compatibility mode, and returns to the task. The task continues execution in compatibility mode.

The VAX-11 system provides compatibility mode capabilities to support the migration of task images from RSX-11M operating systems to VAX/VMS. Compatibility mode provides a framework in which users can run existing task images while upgrading to take full advantage of VAX/VMS native capabilities.

The VAX-11 system also provides facilities for developing RSX-11M tasks. See the VAX-11/RSX-11M User's Guide.

Compatibility mode programs requiring floating-point instruction emulation do not run as fast under VAX/VMS as on a PDP-11. Compatibility mode programs not requiring floating-point emulation and written for a PDP-11/70 run under VAX/VMS at approximately the same speed as they do under the system for which they were written. Programs not requiring floating-point emulation and written for smaller PDP-11 processors run faster under VAX/VMS.

For Version 3.0, VAX/VMS supports execution of RSX-11M Version 3.2 nonprivileged task images in compatibility mode. The majority of nonprivileged, user mode RSX-11M Version 3.2 task images run under VAX/VMS without program modification or rebuilding. Others require modification.

VAX/VMS provides the RSX-11M components (for example, MACRO-11 and the RSX-11M Task Builder) needed to make required modifications using the VAX/VMS system as host. Modifications also can be made using an RSX-11M system.

For an RSX-11M task image to execute successfully under VAX/VMS, it must adhere to the requirements for compatibility mode operation. Both the VAX-11 and VAX/VMS define specific requirements. These requirements are detailed in Sections 1.1 and 1.2.

1.1 VAX-11 COMPATIBILITY WITH PDP-11 PROCESSORS

VAX-11 compatibility mode supports PDP-11 user mode operations. That is, any PDP-11 program that operates only in user mode (not in PDP-11 supervisor or kernel mode) and assumes that I-space and D-space are overmapped potentially can run in VAX-11 compatibility mode.

Any instruction or operation denied to a user mode program executing on a PDP-11 is not allowed in VAX-11 compatibility mode. For example, use of privileged instructions such as HALT and RESET is not permitted; an attempt to use a privileged instruction causes a trap to VAX/VMS, and a subsequent error message.

The VAX-11 compatibility mode instruction set does not support the FIS or FPP floating-point instructions, but VAX/VMS emulates the FPP instructions. Appendix A of this document lists the VAX-11 compatibility mode instruction set.

VAX/VMS places further conditions on the use of the hardware by RSX-11M task images running in compatibility mode. These conditions are detailed in Section 1.2.

1.2 VAX/VMS COMPATIBILITY WITH RSX-11M VERSION 3.2

VAX/VMS supports the capabilities of RSX-11M Version 3.2 to allow the execution of RSX-11M task images. However, to run in compatibility mode, a task image must meet the following requirements.

- It must adhere to the hardware requirements for compatibility mode.
- It must have been built by the RSX-11M Task Builder, Version 3.1 or later, or by the Task Builder (the DCL command LINK/RSX11) provided with VAX/VMS Version 1.0 or later.

- It must be executable in a mapped RSX-11M system. An exception to this rule is a task built as a "multi-user" task. Such tasks cannot be run under RSX-11M. See Section 1.4.

- It must not depend on environmental features of RSX-11M that are not available in VAX/VMS, such as partitions, PLAS, or significant events. Environmental differences between RSX-11M and VAX/VMS are discussed further in Chapters 2 and 3 of this document.

- It must execute within the limitations of task/executive interaction described in the RSX-11M/M-PLUS Executive Reference Manual. It must not be privileged for the purpose of overmapping the RSX-11M executive. The RSX-11M executive is not present in a VAX/VMS system.

- It must not overmap the I/O page. The PDP-11 I/O page is not present in VAX-11 hardware.

- It must not depend on the 32-word memory granularity of the KT11 memory management unit.

- The image must not use any DECnet-11 functions.

RSX-11M task images must not depend on special memory management features available to RSX-11M privileged tasks. Tasks can, however, perform privileged functions that do not involve mapping of the executive. For example, a task executing in compatibility mode can use the QIO$ function codes IO.RLB and IO.WLB to read directly from and write directly to a mounted volume if the system manager has not restricted the user from so doing.

Task images developed under RSX-11M-PLUS, RSX-11D, or IAS that are compatible with RSX-11M Version 3.2 can execute under VAX/VMS if they meet the requirements listed above. However, RSX-11D or IAS task images must be rebuilt using the RSX-11M Task Builder, Version 3.2 or later. RSX-11M task images do not have to be rebuilt to run under VAX/VMS unless program modification or different Task Builder options are required. Rebuilding is also required to take advantage of the logical name extensions of FCS and RMS-11 which allow processing of VAX/VMS file specifications.

1.3 RSX-11M DIRECTIVE REQUESTS

In RSX-11M, a task image interfaces with the operating system by issuing directive requests. As a result of a directive request, RSX-11M performs the desired function and returns control to the task.

VAX/VMS duplicates this task/system interaction. When an RSX-11M task image issues a directive, the hardware traps to VAX/VMS. VAX/VMS duplicates the requested RSX-11M function with either of the following results.

- The RSX-11M directive function is duplicated in VAX/VMS, and the image continues execution.

- VAX/VMS cannot duplicate the requested function but does take whatever action is necessary to allow the task to continue execution.

VAX/VMS duplicates the functions of most RSX-11M directives. When VAX/VMS cannot duplicate an RSX-11M directive, it is because of differences in the basic concepts of the two systems, that is, differences in the environments of the two systems.

For example, the RSX-11M capability to declare a significant event does not exist in VAX/VMS; therefore, VAX/VMS cannot declare a significant event when it receives such a directive request. Rather, it performs no operation and returns a success status to the requesting task image, which continues execution normally.

Subsequent chapters of this document describe the details and implications of directive handlng in VAX/VMS.

1.4 RSX-11M MEMORY HANDLING OPTIONS

VAX/VMS supports the use of overlays produced using the overlay descriptor language of the RSX-11M task builder by RSX-11M images. VAX/VMS loads overlays from the image file at the appropriate point in image execution.

VAX/VMS also supports use of shared regions by RSX-11M images. RSX-11M images can access both shared commons and libraries. Permanently available shared regions are identified to VAX/VMS by the system manager, as described in the VAX/VMS System Management and Operations Guide. Temporary regions are dynamically loaded when an image requiring them executes.

In addition, VAX/VMS supports multiuser (shareable) task images. That is, when a task image is specified at task build time as consisting of a shareable and nonshareable portion, VAX/VMS allows multiple users to access the shareable portion simultaneously. Each user has a private copy of the nonshareable portion. Note that such task images cannot be run under RSX-11M.

VAX/VMS does not support the RSX-11M memory management directives that extend the program logical address space (PLAS) of an RSX-11M task. Any task image issuing a memory management directive under VAX/VMS receives an error status return.

1.5 EMULATION OF FLOATING-POINT INSTRUCTIONS

VAX/VMS provides software emulation of the PDP-11 FPP floating-point instructions for images running in compatibility mode. The time required for emulation of an ADDF (register to register) or ADDF (memory to register) is approximately 25 times that required on a PDP-11/70. This timing is typical of most FPP instructions emulated.

Results produced during emulation are the same as those produced by PDP-11 processors with the following two exceptions:

1. The result of a MOD instruction is more accurate under emulation.

2. On overflow, the emulator generates a reserved operand with a value of 0, rather than providing the residue.

The FPP instruction set is detailed in the PDP-11/04/34A/44/70 Processor Handbook.

1.6 VAX/VMS SYSTEM CONCEPTS

In VAX/VMS, the concept of an RSX-11M task is separated into its two basic components: (1) the program image that executes, and (2) the control information and virtual address space required for image execution. These two components correspond to the VAX/VMS concepts of image and process, respectively. The concepts of image and process are basic to VAX/VMS.

A process is the basic schedulable program entity that the VAX-11 processor executes. A process consists of a virtual address space and control information that both the hardware and software require, such as saved register contents and status information. This control information is called the process context.

An image is the result of linking one or more object modules together. An image can be linked by the VAX-11 linker to execute in native mode or by the RSX-11M Task Builder to run in compatibility mode. An image executes in the virtual address space provided by a process and under control of the process.

A process's virtual address space is divided into two areas: the program region and the control region as illustrated in Figure 1-1. Essentially, the program region provides virtual memory for an image. The control region contains information required by the system to control the process.

The concept of a process with an image is similar to the RSX-11M concept of a task. The main difference is that a task is a form of process that executes a specific image while a process can execute any image. Furthermore, a process remains to execute subsequent images when the current image exits.

In RSX-11M, a reference to a specific task also implies a specific reference to an image. This is not the case with VAX/VMS; therefore, it is useful to state explicitly whether an operation affects a process or its image. For this reason, the terms process and image are used throughout this document instead of the term task.

1.7 DISTINCTION BETWEEN PROGRAMMING AND SYSTEM ENVIRONMENTS

VAX/VMS provides RSX-11M images with an environment similar to that provided by RSX-11M. That is, when an RSX-11M image is loaded, it has a virtual address space starting at location 0. It has access to a copy of its task header in the usual place, and R0 through R7 are initialized as they are under RSX-11M. This environment allows the creation, assembly or compilation, linking, execution, and debugging of RSX-11M images. VAX/VMS does not, however, attempt to duplicate the total environment of the RSX-11M operating system.

For information about developing RSX-11M tasks on VAX/VMS, see the VAX-11/RSX-11M User's Guide.

Certain aspects of the RSX-11M environment have direct equivalents in the VAX/VMS environment. An RSX-11M task name, for example, can be considered a VAX/VMS process name; and RSX-11M event flags can be translated to VAX/VMS event flags.

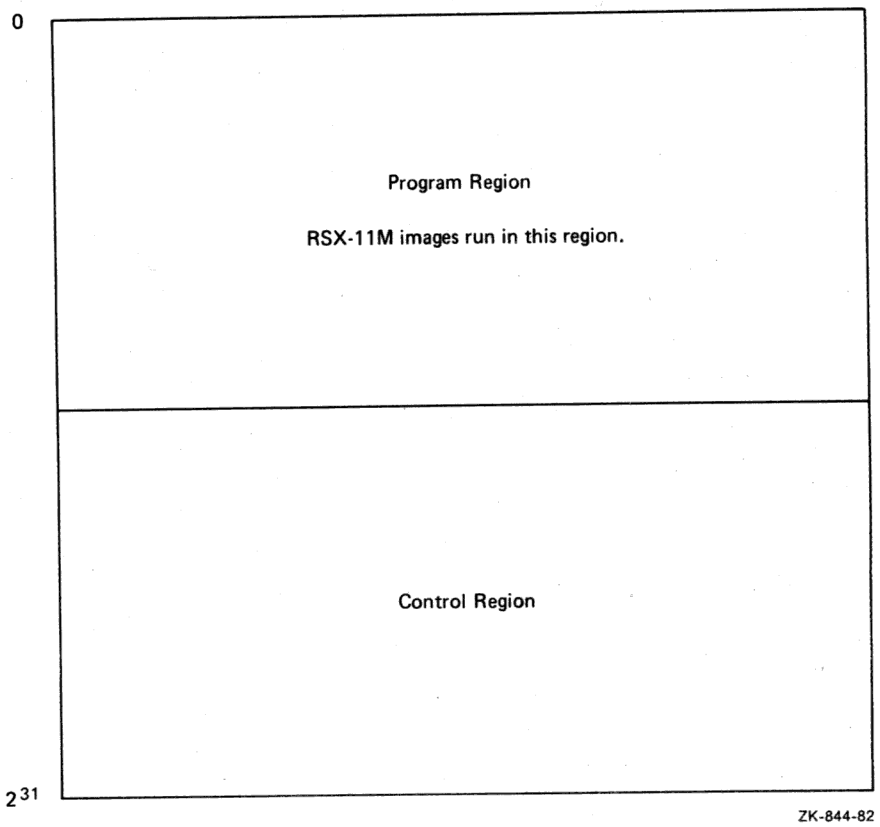


Figure 1-1: Process Virtual Address Space

When a VAX/VMS process executes an RSX-11M image, VAX/VMS receives the traps and exceptions caused by that image and interprets them as RSX-11M would. VAX/VMS then makes a response that is appropriate for the VAX/VMS environment. For example, I/O and send/receive data operations become appropriate VAX/VMS functions.

Other aspects of RSX-11M equate to similar VAX/VMS functions. For example, both systems use user identification codes (UICs). In RSX-11M, UICs are account (login) identifiers, provide a default user file directory (UFD), and are used for file protection. In VAX/VMS, the UIC concept is separated from those of account identifier and default directory name. Rather, the UIC concept is expanded in the direction of additional protection, as described in Section 2.2.

Finally, some aspects of RSX-11M have no counterpart in VAX/VMS. Because no parallel function exists in VAX/VMS, VAX/VMS cannot translate certain aspects of RSX-11M to VAX/VMS functions. Examples of RSX-11M system environment aspects not emulated under VAX/VMS are partitions, significant events, and a range of priorities from 1 through 250. Although VAX/VMS does not duplicate these RSX-11M features, it does accept image requests related to them and takes an appropriate action to allow image execution to continue.

CHAPTER 2

THE VAX/VMS SYSTEM ENVIRONMENT

The environment that VAX/VMS provides for an RSX-11M image is determined by two factors:

1. The privileges, UICs, and resource usage limits allotted to the user who initiates the image

2. The VAX/VMS system components and conventions used to support RSX-11M directives requested by the image

2.1 PRIVILEGES

The system manager maintains a user authorization file that contains an entry for each user. That entry includes a list of the privileges allowed that user's process. All of the privileges that can be associated with a process are described in the VAX-11/RSX-11M User's Guide.

VAX/VMS returns the RSX-11M DSW return code IE.PRI to an RSX-11M image requesting a function for which it does not have the appropriate privilege. The individual directive descriptions in Chapter 4 indicate the DSW codes returned for each directive.

2.2 UIC-BASED PROTECTION

In VAX/VMS, each process has an associated UIC. The UIC consists of 32 bits (1 longword). The member code is in bits 0 through 15 and the group code is in bits 16 through 31, as illustrated in Figure 2-1.

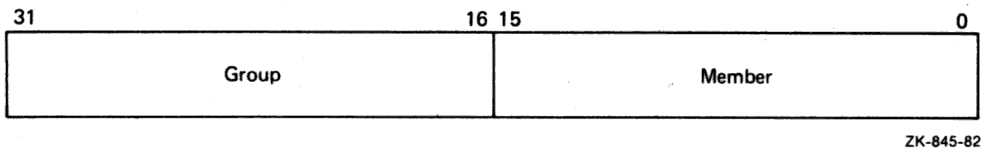


Figure 2-1: VAX/VMS UIC Format

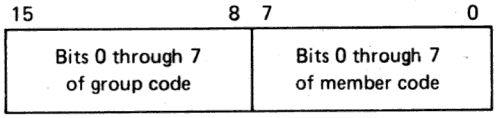
VAX/VMS uses a process's UIC, with the privileges assigned to it by the system manager, to control access to the system services that affect other processes in the system.

When an RSX-11M image issues a directive, VAX/VMS executes the corresponding system service. If this service is restricted by UIC-based protection in VAX/VMS, the group number and privileges of the process executing the RSX-11M image are checked before the service is completed. An error status is returned if the issuing process is not in the appropriate group or does not have the appropriate privilege to affect the target process. ABORT TASK is an example of an RSX-11M directive restricted by UIC-based protection in VAX/VMS.

VAX/VMS ignores the UIC specified when an RSX-11M image is built.

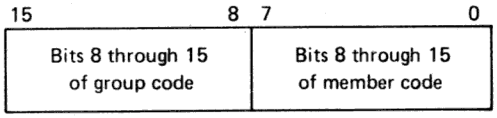
An RSX-11M image can gain access to the UIC of its process by issuing a GET TASK PARAMETERS directive. The UIC is returned in two words of the GET TASK PARAMETERS buffer:

- The low-order byte of the group code and the low-order byte of the member code are returned in word 7, as under RSX-11M:



ZK-846-82

- The high-order byte of the group code and the high-order byte of the member code are returned in word 15:



ZK-847-82

The UIC returned is for informational purposes only. The RSX-11M image cannot use it to affect group protection or file protection.

An RSX-11M image cannot assume that its default account name is related to its UIC. VAX/VMS provides a special directive that is used by File Control Services (FCS) and RMS-11 to access the actual account name.

2.3 RESOURCE USAGE LIMITS

VAX/VMS controls a process's use of system resources by enforcing usage limits defined in the user's authorization file entry. All of the limits that can be defined for a process are described in the VAX-11/RSX-11M User's Guide. The following quotas may be relevant to an RSX-11M image running in VAX/VMS.

- Number of active buffered I/O requests
- Number of bytes of system dynamic memory used for buffered I/O
- Number of active direct I/O requests
- Number of files open simultaneously
- Disk quotas

By default, VAX/VMS places an RSX-11M or native image that attempts to exceed a resource limit in a wait state until the function can be accomplished without exceeding the limit (for example, until other active I/O requests have completed). Native images can disable and enable resource waiting. The DCL and MCR RUN commands provide an option for controlling resource wait mode for subprocesses and detached processes.

If an RSX-11M image attempts to exceed a limit when resource waiting is disabled, the image receives a Directive Status Word (DSW) code of IE.UPN (insufficient dynamic memory).

2.4 PROCESS NAMES

Each process in a VAX/VMS system has a unique 32-bit process identification and a process name. A process name is qualified by its UIC and is unique within a system.

When a user initiates an RSX-11M image that has a task name in its image label block (that is, a task name specified at build time), VAX/VMS assigns the task name as the process's name during image initialization. That name remains in effect until the image exits. Then, VAX/VMS restores the process name used prior to execution of the RSX-11M image. Because VAX/VMS does not incorporate the concept of an installed task, an RSX-11M image cannot acquire a task name by any means other than task building.

An RSX-11M image must have a task name in its label block to provide a name for its process if any of the following can occur:

- The image is to receive data using the RECEIVE DATA, RECEIVE DATA OR EXIT, or RECEIVE DATA OR STOP directives.
- The image is to cooperate with other images using event flags.
- The process containing the image is to be the target of directive action; for example, the process is to be requested or resumed.

Each of the following RSX-11M directives accepts a task name as an argument.

- ABORT TASK
- CANCEL TIME BASED INITIATION REQUESTS
- REQUEST TASK
- RESUME TASK
- RUN TASK
- SEND DATA
- SPAWN TASK

VAX/VMS supports the RSX-11M convention of naming multiuser MCR tasks with a string that starts with an ellipsis, for example, ...PIP. When VAX/VMS encounters an image with a task name of this type, it recognizes that the image can be run by more than one user simultaneously. For such images, VAX/VMS does not create a process name from the task name or set up the mechanisms for it to receive data from other processes and to synchronize with other processes using event flags.

If an RSX-11M image is to issue directives that specify a process executing a native image as the target, the user must be aware of the difference in the allowable lengths of task names and process names.

A task name has a maximum length of six characters. A process name has a maximum length of 15 characters. Therefore, if an RSX-11M image is to refer to a process running a native image, that process's name must not exceed six characters. An RSX-11M image cannot express a process name that exceeds six characters.

A process running a native image can create a subprocess or a detached process, assign it a process name, and designate an image that the process is to execute. Thus, a process can create a named subprocess or detached process that executes an RSX-11M image. Once the process is created, other processes can issue system service requests in native mode or directive requests in compatibility mode that designate the process as the target. The creator of a subprocess always is allowed to affect the subprocess. Other processes and subprocesses must have either group or world privilege to affect the subprocess.

Subprocess and detached process creation are described in the VAX/VMS System Services Reference Manual.

2.5 EVENT FLAG CLUSTERS

An RSX-11M task can have up to three event flag clusters of 32 bits each. VAX/VMS emulation of these event flags is completely transparent to RSX-11M task images. The event flag clusters are:

- Local event flags, numbered 1 through 32
- Common event flags, numbered 33 through 64
- Group global event flags, numbered 65 through 96

Although event flag emulation is transparent to RSX-11M task images, the handling of these flags is important to interactions between native VAX/VMS processes and RSX-11M task images. A native VAX/VMS process cannot associate with any common or group global event flag cluster outside its UIC group.

Local event flags (flags 1 through 32) are visible only to the local task image. A task image can read and set its local event flags.

Every task image is associated with its local event flag cluster. A native VAX/VMS process cannot associate with the local event flag cluster for an RSX-11M task image.

Common event flags (flags 33 through 64) are visible to all processes associated with the flag cluster. Any process can read and set flags in its associated common event flag cluster.

The name of a common event flag cluster is RSXCOMEFN. VAX/VMS qualifies this name with a UIC group number to protect the flags from processes with different UIC group numbers. Only processes with the correct UIC group number can associate with the common event flag cluster RSXCOMEFN for that UIC group.

An RSX-11M task image is associated with a common event flag cluster only if its task image label block has a task name. Otherwise, the task image has no common event flags.

A native VAX/VMS process can associate with the common event flags for an RSX-11M task image. To do this, the process must issue the Associate Common Event Flag Cluster system service, giving an event flag number in the range 64 to 95 and giving the cluster name

RSXCOMEFN. Note that the native process sees the common event flags as flags 64 through 95; an RSX-11M task image sees them as flags 33 through 64.

Group global event flags (flags 65 through 96) are visible to all task images associated with the flag cluster. Any process that is associated with a group global event flag cluster can read and set flags in that cluster.

The name of a group global event flag cluster is RSXGROUPEFN. VAX/VMS qualifies this name with a UIC group number to protect the flags from processes with different UIC group numbers. Only processes with the correct UIC group number can associate with the group global event flag cluster RSXGROUPEFN for that UIC group.

An RSX-11M task image is associated with a group global event flag cluster only if it has issued a CREATE GROUP GLOBAL EVENT FLAGS directive for the cluster. Otherwise, the task image has no group global event flags.

A native VAX/VMS process can associate with a group global event flag cluster for an RSX-11M task image. To do this, the process must issue the Associate Common Event Flag Cluster system service, giving an event flag number in the range 96 to 127 and giving the cluster name RSXGROUPEFN. Note that the native process sees the common event flags as flags 96 through 127; an RSX-11M task image sees them as flags 65 through 96.

In summary, event flag conversion for a VAX/VMS process is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 through 32	32 through 63
Common	33 through 64	64 through 95
Group global	65 through 96	96 through 127

2.6 SYSTEM STATUS CODES

In VAX/VMS, the symbolic name for a system service status return has the following format:

```
SS$_name
```

When an image issues an RSX-11M directive, VAX/VMS attempts to emulate the desired function and then returns a DSW code to indicate success or failure to the image. In most cases, VAX/VMS calls the system service that performs the equivalent of the requested RSX-11M function and converts the status code returned by the service to the equivalent RSX-11M DSW code. For example, the VAX/VMS code SS$_NORMAL becomes DSW code IS.SUC.

In some cases, however, a directive request results in a VAX/VMS error for which no exact RSX-11M equivalent exists. This situation occurs when an image attempts to violate a VAX/VMS concept that has no RSX-11M equivalent. VAX/VMS handles the situation in one of the following ways:

- VAX/VMS returns a default DSW code.
- VAX/VMS returns a DSW code that is meaningful for the error but that could not be returned for the directive if the image were running under RSX-11M.

Default return codes are used when no clear one-to-one relationship exists between VAX/VMS and RSX-11M codes, for example, a VAX/VMS code that is equally related to two DSW codes.

A new DSW code is returned when a VAX/VMS error has no counterpart in RSX-11M. For example, IE.PRI indicates that the image attempted to issue a directive for which its process does not have the appropriate privilege. For example, the image attempted to resume another process in its group but does not have group privilege.

In some cases after a directive failure, VAX/VMS returns an error code in the DSW that is more meaningful to I/O operations. In these cases, the high-order byte of the DSW contains 0. The DSW codes IE.PRI and IE.DUN (for ASSIGN LUN) are examples of codes that are returned as bytes rather than words. RSX-11M images can determine whether a DSW code is returned as a byte or word by testing the high-order byte of the DSW for 0.

DSW codes that can be returned for each directive are listed in Chapter 4 with the individual directive descriptions.

2.7 MEMORY MANAGEMENT

VAX/VMS memory management facilities control the use of physical memory and virtual memory. VAX/VMS controls the use of physical memory by processes through the implementation of two concepts:

- Balance sets
- Working sets

The VAX/VMS Summary Description and Glossary discusses these concepts.

2.7.1 Balance Sets and Working Sets

VAX/VMS divides the virtual and physical address space into fixed-size memory areas called pages. When a process is created, VAX/VMS assigns to it an initial number of pages that the process can use when it is in physical memory. The pages belonging to all processes in physical memory are referred to as the balance set. Processes in the balance set compete for access to system resources. The total number of pages that each process has in the balance set is called the process's working set.

2.7.2 Functions of the Swapper

In response to a need for additional physical memory resources, the VAX/VMS swapper can reduce the working set size of selected processes. If the demand for additional physical memory is not met by reducing working set sizes, VAX/VMS can write selected working sets to secondary storage (also known as swapping).

VAX/VMS reduces working sets and swaps processes from and to main memory to ensure that the highest priority processes are always available in memory for execution. The VAX/VMS swapper is more sophisticated than the RSX-11M checkpointing function. It does, however, provide an equivalent mechanism to allow emulation of the RSX-11M ENABLE CHECKPOINTING and DISABLE CHECKPOINTING directives.

The initial state of an RSX-11M image in a process is to have swapping (checkpointing) enabled. This state is identical to the initial state of an image under RSX-11M. To use the DISABLE CHECKPOINTING directive, an RSX-11M image must have the VAX/VMS privilege to set its swapping mode.

Because VAX/VMS controls the use of physical memory by reducing working sets and swapping processes out of and into a balance set, it does not support partitioning of physical memory. As a result, when an RSX-11M image issues a GET PARTITION PARAMETERS directive, VAX/VMS returns a standard response for a system-controlled partition named GEN. See Chapter 4 for a description of the GET PARTITION PARAMETERS directive.

VAX/VMS ignores the partition name in the image label block.

2.7.3 Functions of the Pager

The VAX/VMS pager controls the number of pages of a process's virtual address space that are in physical memory (that is, in the working set) at any time during process execution. When a process's working set is full and a page in virtual address space is required, the pager replaces the oldest page in the process's working set with the new page.

VAX/VMS facilities for control of a process's virtual address space differ significantly from the RSX-11M approach to a task's virtual memory. As a result, VAX/VMS does not support the RSX-11M memory management (PLAS) directives.

Every RSX-11M image has 64K bytes (32K words) of virtual memory available to it. Because the address space is virtual rather than physical, RSX-11M images can avoid overlaying; an image executes more efficiently by depending on VAX/VMS memory management to determine which pages are needed in physical memory and when they are needed. Further efficiency can be gained by building RSX-11M images as shareable (/MU). These RSX-11M images can be partially shared under VAX/VMS. Note that the shareable feature itself is not supported by RSX-11M.

2.8 SYSTEM EVENTS

A system event in VAX/VMS is an occurrence that affects the ability of one or more processes in the system to execute. For example, an executing process can put itself into a wait state, or it can set an event flag that makes another process a candidate for execution. System events are similar in concept to RSX-11M significant events. In VAX/VMS, however, an image cannot request the declaration of a system event. No VAX/VMS equivalents for the DECLARE SIGNIFICANT EVENT and WAIT FOR SIGNIFICANT EVENT directives exist. Issuing either of these directives has no effect on VAX/VMS; success status is returned to the issuing image.

Therefore, task images that run under VAX/VMS must use only event flags, mailboxes (see Section 3.8), and Asynchronous System Traps (ASTs) for intertask communication; they cannot meaningfully use significant events.

2.9 SYSTEM CLOCK

On PDP-11 systems, the number of ticks per second varies depending on the type of clock used and its frequency. For the time-related directives, VAX/VMS emulates a 100-tick-per-second clock. This difference may affect emulation of the following directives, which have time-oriented arguments.

- MARK TIME
- RUN
- GET TIME PARAMETERS

The RSX-11M directive, SET SYSTEM TIME, is not supported by VAX/VMS compatibility mode.

2.10 SOFTWARE PRIORITIES

VAX/VMS priorities range from 0 through 15 for normal processes and from 16 through 31 for real-time processes. For further details on VAX/VMS handling of priorities, see the VAX/VMS Summary Description and Glossary.

Because RSX-11M process priorities do not correspond to VAX/VMS priorities in a meaningful fashion, VAX/VMS does not attempt to convert a task's priority, as specified in the image's task header, to a VAX/VMS priority.

An RSX-11M image runs at a priority that is determined by the default priority in the user authorization file entry for the user initiating the process. When an image issues an ALTER PRIORITY directive, VAX/VMS performs no operation, and image execution continues at the original process priority. An image requiring high priority must execute in a process that has sufficiently high priority to meet the image's needs.

2.11 GLOBAL SECTIONS

In VAX/VMS, global sections contain data or code that can be brought into memory and made available to processes for manipulation and execution. Global sections are created by executing images and by the system manager.

When a global section is created, its creator assigns a set of characteristics to it. A global section can have the following characteristics:

- Read-only or read/write
- Temporary or permanent
- Group or system wide
- Disk file or paging file

A temporary global section remains in the system only as long as processes are mapped to it; when no processes are mapped to it, VAX/VMS deletes it automatically. A permanent global section remains in the system until it is explicitly deleted.

VAX/VMS provides group protection for group global sections. Any process can gain access to a system global section. A process must be privileged to create a permanent or system global section.

VAX/VMS imposes no limit on the number of global disk file sections to which a process can map. The system limit of paging file global pages is set during system generation.

A disk file global section is mapped to a disk file, whereas a paging file global section is not mapped to a disk file.

When VAX/VMS loads an RSX-11M task image that was built specifying one of the options COMMON, LIBR, RESCOM, or RESLIB, it sets up the specified library or common for the image. When VAX/VMS loads the RSX-11M image, it determines whether the global section for the library or common already exists.

If the global section exists, it is one of the following:

- A permanent global section created by the system manager
- A temporary global section created by VAX/VMS as a result of previous RSX-11M image execution

In either case, VAX/VMS maps the RSX-11M image to the global section.

If the global section does not exist, VAX/VMS creates a temporary group global section for the library or common specified in the COMMON, LIBR, RESCOM, or RESLIB option to the Task Builder. The image file for either the library or common must be located on logical device and directory SYS$LIBRARY.

When VAX/VMS creates a global section for use by RSX-11M images, the section has the following characteristics:

- Global sections are accessed as either read-only or read/write, and either position-dependent or position-independent, according to the Task Builder specification.
- Global sections are group and temporary.
- The global section name is either the library name specified in a COMMON or LIBR option or the file name specified in a RESCOM or RESLIB option.

The disk file for a read/write global section is updated to reflect data manipulation by processes that map to it.

VAX/VMS does not incorporate the concept of an installed global section that can be reinstalled to obtain a fresh copy. The disk file for a read/write global section is updated to reflect data written in the global section. Therefore, if it is necessary to maintain the original state of a read/write (common) global section, the user must keep a protected copy of the common file in a place other than SYS$LIBRARY.

If the library or common area referred to is not found, VAX/VMS prints an error message on SYS$ERROR specifying the name of the library or common.

2.12 HIBERNATION

A hibernating process is known to the system, but is inactive. Suspending or stopping a task image causes the process executing the task image to hibernate.

A suspended task image can be reactivated by one of the following:

- An asynchronous system trap (AST)
- A REQUEST TASK directive
- A RESUME TASK directive
- A RUN TASK directive

A stopped task image can be reactivated only by one of the following:

- An UNSTOP TASK directive located in an AST service routine
- An UNSTOP TASK directive located in another process
- A specified event flag being set

In both DCL and MCR, RUN command options allow creation of a subprocess or detached process that is initially hibernating (rather than active). Before placing the process into hibernation, VAX/VMS loads the image, assigns any needed devices, and loads any needed libraries and common areas.

2.13 IMAGE TERMINATION

An image running under VAX/VMS can terminate normally or abnormally. Normal termination occurs when the image terminates itself by executing the appropriate directive. Abnormal termination occurs when the system or another process forces the image to exit.

2.13.1 Normal Termination

When an RSX-11M image terminates normally, VAX/VMS performs the same image cleanup operations as it does for a native image. If an RSX-11M image issues a TASK EXIT directive, VAX/VMS executes an Exit system service and returns the termination status of SS$_NORMAL.

RSX-11M images also can issue an EXIT WITH STATUS directive to specify the appropriate status. For both VAX/VMS and RSX-11M images, the termination status is available to the command interpreter.

Both the DCL and MCR command interpreters use the termination status when processing indirect command files. DCL uses the termination status with the ON command for error handling. MCR uses the termination status with .ONERR handling. The VAX-11/RSX-11M User's Guide describes the use of indirect command files with the MCR command interpreter. The VAX/VMS Guide to Using Command Procedures describes the use of DCL command procedures (indirect command files).

2.13.2 Abnormal Termination

When a VAX/VMS image incurs a potentially fatal error condition, either of the following can occur:

- The image can handle the condition
- The image cannot handle the condition so VAX/VMS forces the image to terminate

VAX/VMS images can react to fatal errors using the VAX/VMS condition-handling mechanism. Through that mechanism, an image can provide one or more condition-handling routines that are to be executed to handle an exception (error) condition. The condition-handling mechanism provides a function that is comparable to, but more flexible than, the RSX-11M synchronous system trap (SST) mechanism. VAX/VMS condition handling is described in the VAX/VMS System Services Reference Manual.

If an image incurring an exception handles it, the image can continue execution or exit normally, as described above. If the image does not handle the exception, the system terminates the image by issuing an Exit system service. The Exit system service initiates image-related clean-up operations and saves the termination status. This status is available to the command interpreter or the next image to execute in the process.

Abnormal termination of an RSX-11M image can occur as a result of any of the following:

- Violation of the hardware conventions for images running in compatibility mode
- Issuance of an instruction, other than EMT 377, that causes a trap
- Use of an illegal JMP or JSR instruction format
- Occurrence of an odd address error
- Violation of memory protection
- Request for an abort from another process
- Attempt to exceed virtual memory usage limits

An RSX-11M image can supply a synchronous system trap (SST) service routine to handle some of the errors listed above. If the address of an SST service routine for an error is supplied in the SST vector table and that error occurs, VAX/VMS continues image execution in the SST routine. This routine determines whether the image is to exit or continue. If no SST address is supplied, VAX/VMS terminates the image.

If the error is one that cannot be handled by an SST service routine or if no valid SST routine address is supplied, VAX/VMS issues a termination message on the device assigned to SYS$ERROR and SYS$OUTPUT. VAX/VMS causes the image to exit with a termination status that is available to the command interpreter.

Table 2-1 lists the reasons for image termination and indicates which errors can be handled by an SST service routine. The status codes in parentheses following the termination messages are defined by $RSXDEF macro.

Table 2-1: Reasons for RSX-11M Image Termination

Status Code	Message and Explanation
RSX$_IOT	IOT EXECUTION The image executed an IOT instruction.
RSX$_BREAK	BPT EXECUTION The image executed a BPT instruction.
RSX$_TBIT	T-BIT EXECUTION The image executed an instruction requiring a T-bit trap.
RSX$_TRAP	TRAP EXECUTION The image executed a trap instruction.
RSX$_NONRSXEMT	NON-RSX EMT EXECUTION The image executed an invalid EMT instruction.
RSX$_ILLINST	ILLEGAL INSTRUCTION The image executed a JMP or JSR instruction with a register as the destination.[1]
RSX$_ACCVIO	MEMORY PROTECTION VIOLATION The image addressed a location outside its virtual address space.[1]
RSX$_ODDADDR	ODD ADDRESS ERROR The image addressed a word at an odd address (nonword boundary).[1]
RSX$_RESERVED	RESERVED INSTRUCTION The image executed an instruction that is not allowed in compatibility mode (HALT, MARK, RESET, SPL, or WAIT).[1]
RSX$_BADSTACK	BAD STACK The stack pointer contains an address outside the image's virtual address space.[2]
RSX$_INSFDYNMEM	NO DYNAMIC SPACE A requested service needs more dynamic space than the process is allowed.[2]

1. The returned PC is the address of the invalid instruction, not the address following it.

2. This error cannot be trapped to the SST vector table.

2.14 PARSING OF FILE SPECIFICATIONS

Because of the VAX/VMS logical name capability, VAX/VMS file specifications can differ from those used in RSX-11M. The normal RSX-11M parsing routines cannot provide defaults for such VAX/VMS file specifications. VAX/VMS provides a special directive that is issued by FCS and RMS-11 running under VAX/VMS so that they provide proper defaults in a manner that is transparent to the RSX-11M image. Any RSX-11M image that performs its own parsing also must call this special directive, which is described in Appendix B.

An RSX-11M image issuing such a directive uses the same sources for default information as it does under RSX-11M (for example, the default file name block and directory string). When the directive is issued, VAX/VMS builds the necessary data structures and calls VAX-11 RMS. When VAX-11 RMS returns the expanded file specification, VAX/VMS returns it to the image in the format used by FCS and RMS-11 (for example, in the resultant file name block and directory string).

Compatibility mode software uses negative version numbers for files in a manner consistent with VAX/VMS, not RSX-11M. You can use version number 0 to indicate the most recent version of a file (like RSX-11M), version number -0 to indicate the oldest version (like -1 in RSX-11M), version number -1 to indicate the next to the highest version (unlike RSX-11M), -2 to indicate the second most highest version (unlike RSX-11M), and so forth.

2.15 VAX/VMS I/O SYSTEM

VAX/VMS uses its own I/O system in duplicating RSX-11M I/O operations. Components at all levels of the VAX/VMS I/O system provide functions that are similar to equivalent functions in RSX-11M. For example, VAX/VMS I/O system services provide functions similar to those provided by RSX-11M I/O directives. Differences between the two I/O systems arise in the following cases:

- VAX/VMS implementation of a function varies from RSX-11M implementation to provide more flexibility or efficiency (for example, certain Queue I/O Request function codes)
- VAX/VMS implementation of a function or concept not provided in RSX-11M and use of that function in emulating RSX-11M I/O

Such differences affect emulation of the following I/O-related directives.

- ASSIGN LUN
- GET LUN INFORMATION
- QUEUE I/O REQUEST
- QUEUE I/O REQUEST AND WAIT
- SEND DATA
- RECEIVE DATA
- RECEIVE DATA OR EXIT
- RECEIVE DATA OR STOP

Chapter 3 presents an overview of the VAX/VMS I/O system and relates aspects of it to an RSX-11M image.

CHAPTER 3

VAX/VMS I/O SYSTEM

The VAX/VMS I/O system comprises the following components:

- VAX-11 Record Management Services (VAX-11 RMS) for user-level, device-independent I/O (native mode only)
- I/O system services that provide the means for an image to assign devices and issue I/O requests directly
- Ancillary control processes (ACPs) for performing file-oriented functions on disk and magnetic tape volumes
- I/O drivers

Figure 3-1 illustrates the relationships among these components for native mode programming. If you are already familiar with the VAX-11 native mode programming components, you can skip to Section 3.4.

3.1 VAX-11 RECORD MANAGEMENT SERVICES (VAX-11 RMS)

VAX-11 RMS provides native VAX/VMS images with the capability to perform device-independent I/O. Images issue commands to open a file, get and put records or read and write blocks, and close the file. VAX-11 RMS, in turn, issues the I/O system services that cause the driver or ancillary control process (ACP) to perform the function requested by the user.

VAX-11 RMS is the VAX/VMS equivalent of FCS and RMS-11. It has no direct effect on and is inaccessible to an RSX-11M image executing in compatibility mode. VAX/VMS does, however, call VAX-11 RMS to perform some I/O services on behalf of an RSX-11M image.

VAX-11 RMS is described in the VAX-11 Record Management Services Reference Manual.

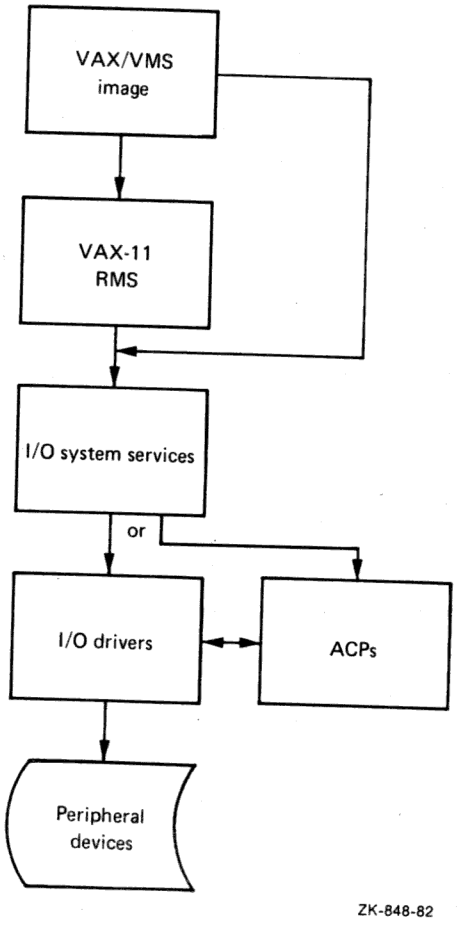


Figure 3-1: Components of VAX/VMS I/O System

3.2 VAX/VMS I/O SYSTEM SERVICES

A native image can call VAX/VMS I/O system services to describe its I/O requirements directly, that is, without using VAX-11 RMS. The request can be issued by a user image or by VAX-11 RMS on behalf of a user image. I/O services allow suitably privileged processes to request the following functions:

- Assign and deassign channels
- Queue an I/O request and (optionally) wait for its completion
- Create and delete mailboxes
- Allocate and deallocate devices
- Mount and dismount volumes or volume sets
- Get device information
- Cancel I/O on a channel

3.2.1 Assign I/O Channel System Service

Before a VAX/VMS image can request an I/O operation, it must establish a path of reference from the process in which it is executing to the device on which the operation is to be performed. In VAX/VMS, this path of reference is obtained by calling the Assign I/O Channel system service. This service returns a channel number (path designator) for the assigned device. The channel number remains valid until the image deassigns the channel or terminates.

In addition to the channels assigned by an image, a process has channels assigned by the system. These channels are permanent for the duration of the process. They provide the path of reference for the process-permanent files used for system input (SYS$INPUT and SYS$COMMAND), system output (SYS$OUTPUT), error messages (SYS$ERROR), and any user-created process-permanent files. For RSX-11M images under VAX/VMS, user-created process-permanent files appear as record-oriented terminal devices.

An image can request I/O operations on channels that it assigns and on those that the system assigns to process-permanent files. However, VAX-11 RMS must be used for I/O operations to process-permanent files except those mapping to terminal devices. The Assign I/O Channel system service is the VAX/VMS equivalent of the ASSIGN LUN directive.

3.2.2 Queue I/O Request System Service

Once the image has assigned a channel to a device, the image can request I/O operations by calling the Queue I/O Request system service and specifying the channel number returned by the Assign I/O Channel system service as an argument. Additional arguments provide function-dependent and function-independent data required for the I/O operation.

When called, the Queue I/O Request system service allocates and builds an I/O request packet that describes the operation to be performed as indicated by the arguments passed to it by the image. Once the packet is built, the Queue I/O Request system service places the packet in a queue of requests for the designated device. Requests are queued according to the priority of the process from which the image issued the request. The driver for the device unit dequeues requests by priority and performs them.

3.2.3 Create Mailbox and Assign I/O Channel System Service

The Create Mailbox and Assign I/O Channel system service lets an image create a virtual device, called a mailbox, and assign an I/O channel to it. Mailboxes provide the mechanism for protected interprocess communication in VAX/VMS. Normally, an image creates a mailbox from which it reads and to which other images in cooperating processes write. Access to the mailbox is restricted using the normal UIC-based protection according to system, owner, group, and world. An image performs I/O operations on a mailbox using VAX-11 RMS $GET and $PUT commands or the Queue I/O Request system service.

A mailbox has no RSX-11M equivalent. However, VAX/VMS does use mailboxes in duplicating RSX-11M send/receive directives. If a logical name is assigned to an existing mailbox, an RSX-11M image can issue I/O requests to the mailbox using the mailbox's logical name. An RSX-11M image cannot create a mailbox directly. Use of mailboxes for send/receive directives is detailed in Section 3.8.1.

3.2.4 Additional I/O System Services

The Allocate Device system service lets an image reserve a device for exclusive use by the process in which the image is executing. The device remains allocated until it is explicitly deallocated or until the process terminates. VAX/VMS automatically allocates any nonshareable device (for example, terminal or card reader) assigned by a process. It does not automatically allocate a shareable device (for example, disk). The concept of device allocation is the VAX/VMS equivalent of the RSX-11M concept of attaching a device.

System services allow users to mount and dismount a volume or volume set. The Mount system service associates a volume with a device and optionally assigns a logical name to that device. The Dismount system service dissassociates this information.

The Get Device/Volume Information system service lets an image obtain the name and characteristics of the device assigned to a particular channel. It is equivalent to the GET LUN INFORMATION directive in RSX-11M.

The Cancel I/O Request system service lets an image cancel all I/O requests pending on the specified channel. It is equivalent to the RSX-11M QUEUE I/O REQUEST directive with a function code of IO.KIL.

I/O system services are described in the VAX/VMS System Services Reference Manual.

3.3 I/O DRIVERS AND ACPS

Using information in the I/O request packet, the I/O driver for the unit to which the request was queued initiates the actual hardware operation that performs the requested function. Once the transfer is initiated, the driver returns control to the Queue I/O Request system service. The service returns the request status to its caller. When the hardware operation completes, the hardware generates an interrupt that causes the driver to be reentered to complete processing of the I/O request.

When the driver completes the I/O request, it issues a software interrupt for the I/O post-processing routine. The I/O post routine sets up the mechanism that causes user-requested I/O completion information to be passed to the image. For example, it fills in the I/O status block and passes information needed to set an event flag or queue an AST, if either is requested.

If the driver cannot perform the request because it requires handling of file-structured volumes, ACP intervention is needed. In that case, the driver queues the request for the appropriate ACP to perform.

3.4 RSX-11M IMAGE INTERFACE TO THE VAX/VMS I/O SYSTEM

RSX-11M images perform I/O by issuing requests to FCS/RMS-11 level or by using the QIO$ directive level. The number of steps required to perform each I/O operation varies depending on the level of the request. Figure 3-2 illustrates the interface between an RSX-11M image and the VAX/VMS I/O system.

Images issuing FCS and RMS-11 requests use the same FCS and RMS-11 routines available in RSX-11M.[1] Some of these routines have been modified to take advantage of VAX/VMS features, such as directory naming and file specification parsing. To take advantage of the modifications, the RSX-11M image must be rebuilt under VAX/VMS using the RSX-11M Task Builder (see the VAX-11/RSX-11 User's Guide). The VAX/VMS modifications are compatible with RSX-11M versions of FCS and RMS-11.

Both FCS and RMS-11 run in compatibility mode under VAX/VMS. When an RSX-11M image issues either an FCS or RMS-11 request, FCS or RMS-11 receives the request and reacts to it in the same mannner as it does when running in RSX-11M. That is, FCS/RMS-11 issues the appropriate RSX-11M QIO$ directive.

From this point, the steps are identical to those taken when any RSX-11M image issues a QIO$ directive:

- The QIO$ directive traps to VAX/VMS.
- VAX/VMS determines whether the QIO$ was to a process-permanent file, such as TI or SYS$OUTPUT. If it is and that device is not a terminal, VAX/VMS issues a VAX-11 RMS $GET or $PUT request. Otherwise, VAX/VMS issues the VAX/VMS $QIO system service request that corresponds to the RSX-11M QIO$.
- Upon completion of the QIO request, VAX/VMS returns the appropriate DSW code to the issuing image.
- Upon completion of the I/O operation, VAX/VMS returns status information in the I/O status block and sets an event flag or declares an AST, if requested.

If the routine to which the DSW code is returned is either FCS or RMS-11, that component, in turn, makes the appropriate status return to the calling image.

3.5 DEVICE ASSIGNMENT

VAX/VMS performs device assignment for RSX-11M images as part of image initialization when the image is loaded. It also performs device assignment during image execution as a result of an ASSIGN LUN directive.

---

1. Because the VAX/VMS Files-11 ACP does not support block locking, RMS-11 block locking across processes is not supported. As a result, RMS-11 does not allow file sharing for write-accessed files of relative and indexed organization under VAX/VMS.

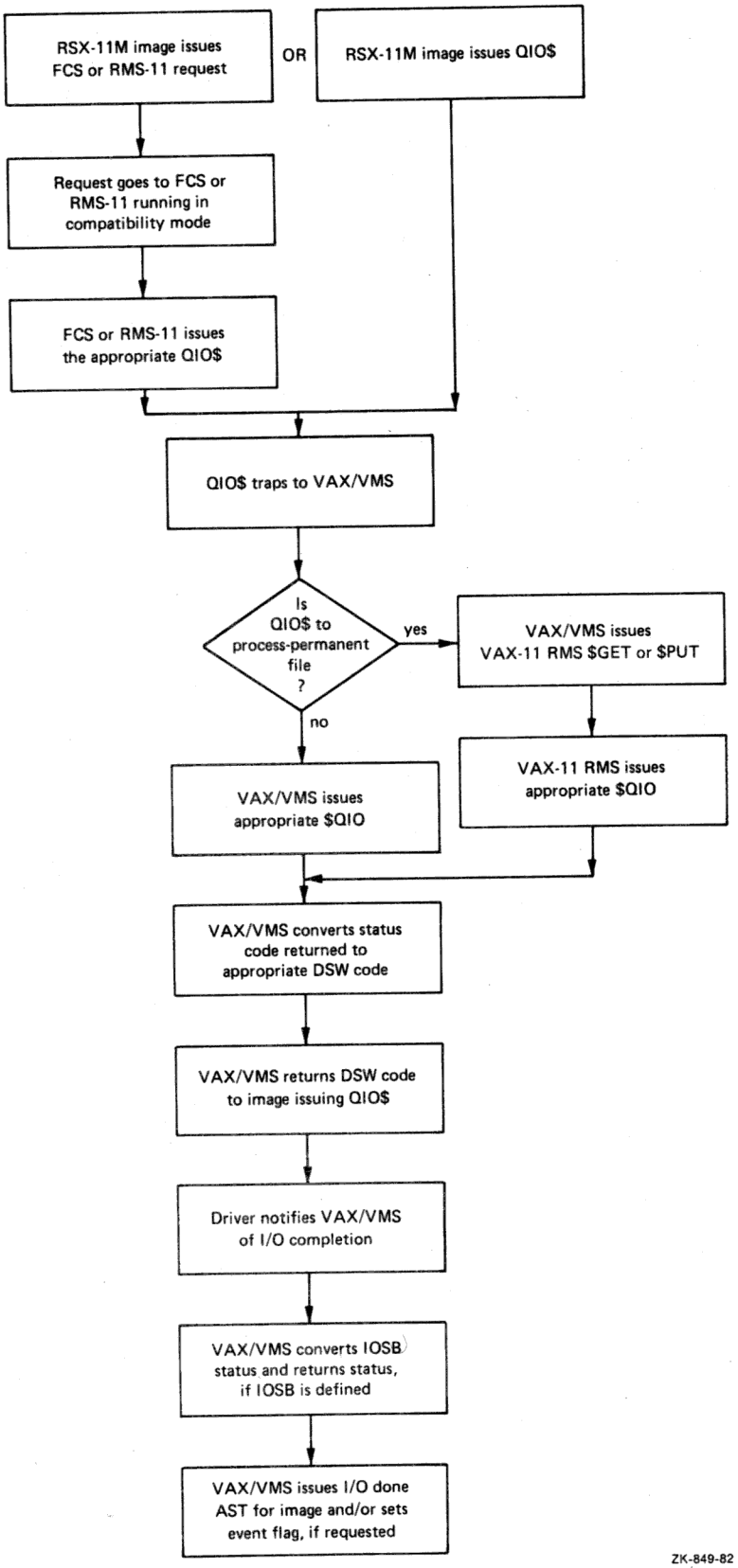


Figure 3-2: RSX-11M Image Interface to VAX/VMS I/O System

In making a device assignment for an RSX-11M image, VAX/VMS proceeds with the following steps, which result in the device unit's physical name.

- VAX/VMS forms an ASCII string using the device name and unit number supplied by the image. VAX/VMS uses the two characters plus the binary unit number supplied. The unit number is converted to ASCII base 8. No editing is performed on the name; for example, if TT1 is supplied, that name is used rather than TT01.

- VAX/VMS attempts to translate the ASCII string as a logical name. If the attempt to translate fails, VAX/VMS assumes that the image supplied an RSX-11M physical device name. VAX/VMS builds a VAX/VMS physical device name using the image's original input and issues an Assign I/O Channel system service using the VAX/VMS device name. VAX/VMS maps RSX-11M physical device names to VAX/VMS physical device names, as described in Section 3.6.

  If the name translates, VAX/VMS attempts up to two more translations. If the maximum number of translations (three) is performed or if one of the attempts results in no translation, VAX/VMS assigns a channel using the final equivalence name.

For example, if IN0 is defined as a process's logical name for TTB3 and that process runs an RSX-11M image, which subsequently issues an ASSIGN LUN directive for IN0, VAX/VMS forms an ASCII string for IN0, translates the string to TTB3, and assigns a channel to TTB3.

3.6 DEVICE MAPPING

If the user does not assign the RSX-11M device name as the logical name for a VAX/VMS physical device unit, VAX/VMS automatically performs the translation to a physical device. (A mailbox is an exception; its device name is MBAn, where n is its unit number.) This conversion is done by converting the RSX-11M unit number to decimal and dividing it by 16 (decimal). The quotient is added to the ASCII value representing the character A. The result is the controller letter. The remainder becomes the VAX/VMS unit number. For example, RSX-11M devices TT0 and DB22 become VAX/VMS devices TTA0 and DBB2, respectively.

Examples:

TT0 to TTA0:

```
Convert octal 0 to decimal = 0
Controller and unit='A'+(0/16)='A'+0 with a remainder of 0

                    'A'+0='A'=controller
                        0=unit number
```

DB22 to DBB2:

```
Convert octal 22 to decimal = 18
Controller and unit='A'+(18/16)='A'+1 with a remainder of 2

                    'A'+1='B'=controller
                        2=unit number
```

VAX/VMS performs this conversion when assigning an I/O device for an RSX-11M image.

To convert back from a VAX/VMS device name to the RSX-11M form, VAX/VMS performs the reverse operation. It subtracts the value representing the ASCII character A (65) from the controller letter and multiplies the result by 16 (decimal). It then adds the VAX/VMS unit number and converts it to octal. The result is an RSX-11M unit number that is appended to the 2-character device name. For example, the VAX/VMS device name LPB1 converts to the RSX-11M device name LP21.

Example:

LPB1 to LP21:

```
Unit =(('B' - 'A') * 16) + 1 = (1 * 16) + 1 = 17 (decimal)
Convert decimal 17 to octal = 21
```

VAX/VMS performs this conversion and stores it in the RSX-11M logical name list for the image. This device information is returned as a result of a GET LUN INFORMATION directive.

3.6.1 Mapping Pseudodevice Names

The pseudodevice names TI, CL, CO, SY, WK, LB, and OV are exceptions to the rules for device name mapping. Table 3-1 shows RSX-11M pseudodevice names that are translated into VAX/VMS logical names.

Table 3-1: Device Name Mapping

RSX-11M Name	GLUN$ Name	VAX/VMS Name
TI	$I0 $O0	SYS$INPUT SYS$OUTPUT
CL	$E0	SYS$ERROR
CO	$C0	SYS$COMMAND
SY	Mapped name	SYS$DISK
SP	Mapped name	Name assigned by system manager
WK	Mapped name	SYS$SCRATCH
LB	Mapped name	SYS$SYSROOT
OV	--	For the LUN used by the overlay run-time system, OV0 translates to provide access to the task image file. Any other LUNs assigned to OV0 cause VAX/VMS to assign the device on which the image resides.

3.6.2 The LB Pseudodevice Name and Concealed Devices

A new type of device has been defined in VAX/VMS Version 3.0. This new device, called a "concealed device," has two underscore characters (__) preceding the physical device name.

Logical name translation is not as tightly bound to actual device names as it was in previous versions of VAX/VMS. The device name parameter provided to a VAX/VMS system service is now fully translated by the system service. In addition, VAX-11 RMS returns, as the device name, a logical name that translates to a concealed device.

A new directory concept was introduced with VAX/VMS Version 3.0 that is associated with concealed devices, called a "root directory." A concealed device can refer to either an actual device or to a device and directory combination, called a "rooted device." Using logical names, a device can be referenced as a rooted device or an actual device, but not both at the same time. When you use a rooted device logical name, such as the system-defined logical name SYS$SYSROOT, only subdirectories of the root directory on that device can be accessed, not the Master File Directory (MFD) and all of its associated first-level directories.

Most applications are not affected by the addition of rooted directories. However, the RSX-11M pseudodevice name LB translates to a rooted device (SYS$SYSROOT) to allow access to system libraries. Initially, you can only reference subdirectories of the root directory using the LB name, such as [1,1] and [1,2]. If an RSX-11M image attempts to access a file in another directory using the LB name as part of the file specification, it will unsuccessfully attempt to access that directory as a subdirectory of the root structure and not the MFD, unless the LB logical name assignment is changed.

If an RSX-11M image uses the LB name, you can use the MCR command ASN (see the VAX-11/RSX-11M User's Guide) to change the LB rooted device logical name to a logical name that translates to an actual device. However, attempts to access system library directories while LB is assigned to an actual device will fail. Simply deassign the LB name to reset it to its initial translation when needed.

For additional information on concealed devices, see the VAX-11 Record Management Services Reference Manual.

3.7 HANDLING OF QUEUE I/O FUNCTION CODES

VAX/VMS provides both device-independent and device-dependent functions at the Queue I/O Request service level. Device-independent functions include read and write virtual block, read and write logical block, and read and write physical block. Device-dependent functions include operations such as the handling of control and escape sequences for terminal I/O and positioning functions for magnetic tape.

For most RSX-11M function codes, VAX/VMS has a corresponding function code or system service. All disk function codes and most magnetic tape function codes have corresponding functions in VAX/VMS. However, two areas exist where discrepancies between RSX-11M and VAX/VMS device handling may appear:

- Handling of terminal devices
- Handling of spooled devices

Details concerning VAX/VMS handling of all RSX-11M device function codes are provided in Chapter 5. The implications of spooling for RSX-11M images is described in Section 3.10.

3.8 MAILBOXES

A mailbox is a record-oriented virtual device used in VAX/VMS for generalized communication among processes. VAX/VMS uses a mailbox to duplicate the RSX-11M RECEIVE DATA, RECEIVE DATA OR EXIT, RECEIVE DATA OR STOP, and SEND DATA directives. These directives are the normal means of intertask communication in the RSX-11M environment.

A mailbox has UIC-based protection associated with it. The creator of the mailbox can specify read and write privileges for system, owner, group, and world. Because the concepts of execute and delete are not meaningful for mailboxes, the creator does not specify privileges for these functions.

When VAX/VMS creates a mailbox for emulating the send/receive directives, it specifies read access for the owner and write access for the group. The owner is the image issuing the receive directives and the group comprises the images issuing the send directives. Owner and group are identified by the UIC under which they execute.

3.8.1 Mailboxes for Send/Receive Directives

When VAX/VMS loads a compatibility mode image, it determines whether the image has a task name by examining the image's task label block. The presence of a task name in the label block is an indication that the image can issue RECEIVE DATA, RECEIVE DATA OR EXIT, and RECEIVE DATA OR STOP directives to obtain data sent to it by other images. The system defines a process name that is identical to the task name in the label block. If the name is unique, just prior to actual image execution, the system creates a mailbox and associates it with the process. The mailbox is named as follows:

RCVDtaskname

The name is qualified by group number. Other images that send data to the mailbox must be within the same group and have group or world privilege.

VAX/VMS does not create a mailbox for an image having a task name in the form ...xxx, for example, ...MAC.

Figure 3-3 illustrates the use of mailboxes for the send and receive functions.

3.8.2 I/O to Mailboxes

A mailbox has a device name of MBAn. The value of n is the unit number. VAX/VMS unit numbers are 5-digit numbers in the range 0 to 65535. When an image creates a mailbox, VAX/VMS assigns a unit number to it. Each time an image executes, the unit number assigned by VAX/VMS to any mailboxes that the image creates can vary.

Because mailboxes are treated as devices under VAX/VMS, any RSX-11M image can assign a channel to a mailbox using its logical name and perform record I/O to it. The RSX-11M image must use the logical name rather than the device name (MBAn) to refer to the mailbox because RSX-11M images can accept only a unit number in the range 0 to 255.

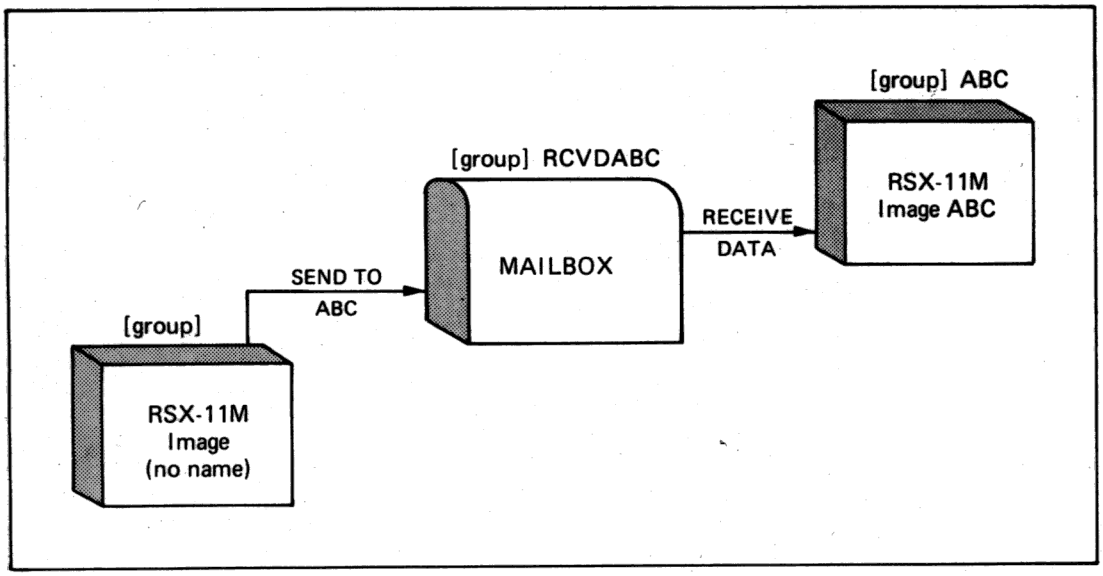


Figure 3-3: Use of Mailboxes for Send/Receive Directives

Either an RSX-11M image or a native VAX/VMS image can assign a mailbox; only a native VAX/VMS image can create a mailbox. A mailbox assigned by an RSX-11M image must be either permanently available in the system or created by a native image. Assignment of a mailbox is treated the same as the assignment of other VAX/VMS devices for RSX-11M images.

A mailbox can be shared by native images and RSX-11M images. As a result, mailboxes provide a convenient means for native images to communicate with RSX-11M images. The mailbox used for such communication can be created by a native image or created by VAX/VMS for emulating the send and receive directives.

A native image can send messages to a mailbox created for directive emulation by issuing write requests to it. The image can use either VAX-11 RMS or the Queue I/O Request system service for the I/O operations.

3.9 ACP FUNCTIONS

RSX-11M Files-11 ACP functions correspond directly to VAX/VMS Files-11 ACP functions. The mapping is transparent to the RSX-11M image, as described in Chapter 5.

3.10 SPOOLED DEVICES

Under VAX/VMS, spooling occurs as a result of cooperation among the I/O related system services: Files-11 ACP, VAX-11 RMS, and output symbionts. Spooling in RSX-11M requires interaction with the RSX-11M spooler. Use of VAX/VMS spooled devices is transparent to RSX-11M images.

If an image assigns a device that is spooled (for example, a line printer) the resulting assignment is actually to an intermediate device (for example, a disk). If the image issues a GET LUN INFORMATION directive, the system returns characteristics that are consistent with the intermediate device containing the spooled files. Characteristics of the final output device (the line printer) are not returned to the RSX-11M image. RSX-11M-PLUS handles spooled device assignment the same way.

If an image uses RMS-11 or FCS to access a spooled device, the file is spooled when it is deaccessed.

Use of the QUEUE I/O REQUEST directive to a VAX/VMS spooled device without preceding the request with an OPEN$ macro or appropriate ACP functions results in a privilege violation status return. Because the device to which the QUEUE I/O REQUEST directive actually is directed is a file-structured device, the appropriate ACP functions (for example, access file) must occur before I/O to the device can be performed. Use of RMS-11 or FCS PUT$ requests ensures that the ACP functions occur.

3.10.1 FCS Spooling

The FCS spooling macro PRINT$ and the services associated with it under RSX-11M are supported in VAX/VMS. Spooling in RSX-11M is accomplished by a task named PRT... . When VAX/VMS detects a SEND DATA directive with PRT... as the target, it executes a Send Message to Symbiont Manager system service to spool the file.

CHAPTER 4

DIRECTIVE DESCRIPTIONS

This chapter describes how VAX/VMS handles RSX-11M Version 3.2 directives. Section 4.1 summarizes differences between VAX/VMS and RSX-11M handling of directives. All directives supported under VAX/VMS are described in Section 4.2. These directives are alphabetized by macro name. VAX/VMS handling that is different from RSX-11M handling is gray-shaded.

NOTE

Directives added since RSX-11M Version 3.2 are not supported.

4.1 VAX/VMS HANDLING OF DIRECTIVES

Table 4-1 outlines the differences between RSX-11M and VAX/VMS handling of directives. For each directive, the table gives the macro name and the principal differences in handling. The numbers in parentheses refer to sections (other than the directive description in this chapter) that further clarify the VAX/VMS handling.

Table 4-1: VAX/VMS Handling of Directives

Macro	Directive Name, Differences, and Section References
ABRT$	ABORT TASK
	Process protected by group (2.2); group or world privilege required (2.1); process name required for target image (2.4)
ALTP$	ALTER PRIORITY
	No operation (2.10)
ALUN$	ASSIGN LUN
	Logical name translated (3.5); device assignment required (3.5); devices mapped (3.6)
ASTX$S	AST SERVICE EXIT
	No differences

(continued on next page)

Table 4-1 (Cont.): VAX/VMS Handling of Directives

Macro	Directive Name, Differences, and Section References
ATRG$	ATTACH REGION
	Not supported
CINT$	CONNECT TO INTERRUPT VECTOR
	Not supported
CLEF$	CLEAR EVENT FLAG
	No differences
CMKT$	CANCEL MARK TIME REQUESTS
	Process protected by group (2.2); GROUP or WORLD privilege required (2.1)
CNCT$	CONNECT
	Not supported
CRAW$	CREATE ADDRESS WINDOW
	Not supported
CRFG$	CREATE GROUP GLOBAL EVENT FLAGS
	Group global event flags protected by group (2.5)
CRRG$	CREATE REGION
	Not supported
CSRQ$	CANCEL TIME BASED INITIATION REQUESTS
	Process protected by group (2.2); GROUP or WORLD privilege required (2.1); process name required for target image (2.4)
DECL$S	DECLARE SIGNIFICANT EVENT
	No operation performed (2.8)
DSAR$S	DISABLE AST RECOGNITION
	No differences
DSCP$S	DISABLE CHECKPOINTING
	Set swap mode privilege (PSWAPM) required (2.1, 2.7.1)
DTRG$	DETACH REGION
	Not supported

(continued on next page)

Table 4-1 (Cont.): VAX/VMS Handling of Directives

Macro	Directive Name, Differences, and Section References
ELAW$	ELIMINATE ADDRESS WINDOW
	Not supported
ELGF$	ELIMINATE GROUP GLOBAL EVENT FLAGS
	Group global event flags protected by group (2.5)
ENAR$S	ENABLE AST RECOGNITION
	No differences
ENCP$S	ENABLE CHECKPOINTING
	Set swap mode privilege (PSWAPM) required (2.1, 2.7.1)
EXIF$	EXIT IF
	Image termination (2.13)
EXIT$S	TASK EXIT
	Image termination (2.13)
EXST$	EXIT WITH STATUS
	Image termination (2.13)
EXTK$	EXTEND TASK
	No differences
GLUN$	GET LUN INFORMATION
	Intermediate device information given for spooled device (3.10, 5.2)
GMCR$	GET MCR COMMAND LINE
	No differences
GMCX$	GET MAPPING CONTEXT
	Not supported
GPRT$	GET PARTITION PARAMETERS
	Parameters given for GEN partition (2.7.1)
GREG$	GET REGION PARAMETERS
	Not supported
GSSW$S	GET SENSE SWITCHES
	Not supported

(continued on next page)

Table 4-1 (Cont.): VAX/VMS Handling of Directives

Macro	Directive Name, Differences, and Section References
GTIM$	GET TIME PARAMETERS
	100-tick-per-second clock used (2.9)
GTSK$	GET TASK PARAMETERS
	Parameters given for GEN partition (2.7.1)
MAP$	MAP ADDRESS WINDOW
	Not supported
MRKT$	MARK TIME
	100 tick-per-second clock used (2.9)
QIO$	QUEUE I/O REQUEST
	Function codes mapped to VAX/VMS function codes (Chapter 5)
QIOW$	QUEUE I/O REQUEST AND WAIT
	Function codes mapped to VAX/VMS function codes (Chapter 5)
RCST$	RECEIVE DATA OR STOP
	Mailbox used (3.8.1); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
RCVD$	RECEIVE DATA
	Mailbox used (3.8.1); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
RCVX$	RECEIVE DATA OR EXIT
	Mailbox used (3.8.1); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
RDAF$	READ ALL EVENT FLAGS
	No differences
RDXF$	READ EXTENDED EVENT FLAGS
	No differences
RQST$	REQUEST TASK
	Active or hibernating target image required (2.12); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)

(continued on next page)

Table 4-1 (Cont.): VAX/VMS Handling of Directives

Macro	Directive Name, Differences, and Section References
RREF$	RECEIVE BY REFERENCE
	Not supported
RSUM$	RESUME TASK
	Protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
RUN$	RUN TASK
	Active or hibernating target image required (2.12); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
SDAT$	SEND DATA
	Mailbox used (3.8.1); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
SETF$	SET EVENT FLAG
	No differences
SFPA$	SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST
	No differences
SPND$S	SUSPEND
	Process name required for caller (2.4)
SPRA$	SPECIFY POWER RECOVERY AST
	No differences
SPWN$	SPAWN
	For command line handling, mailbox used (3.8.1); protected by group (2.2); privilege required (2.1)
SRDA$	SPECIFY RECEIVE DATA AST
	No differences
SREF$	SEND BY REFERENCE
	Not supported
SRRA$	SPECIFY RECEIVE-BY-REFERENCE AST
	Not supported
STLO$	STOP FOR LOGICAL OR OF EVENT FLAGS
	No differences

(continued on next page)

Table 4-1 (Cont.): VAX/VMS Handling of Directives

Macro	Directive Name, Differences, and Section References
STOP$S	STOP
	No differences
STSE$	STOP FOR SINGLE EVENT FLAG
	No differences
SVDB$	SPECIFY SST VECTOR TABLE FOR DEBUGGING AID
	No differences
SVTK$	SPECIFY SST VECTOR TABLE FOR TASK
	No differences
UMAP$	UNMAP ADDRESS WINDOW
	Not supported
USTP$	UNSTOP TASK
	Active or hibernating target image required (2.12); protected by group (2.2); privilege required (2.1); process name required for target image (2.4)
WSIG$S	WAIT FOR SIGNIFICANT EVENT
	No operation performed (2.8)
WTLO$	WAIT FOR LOGICAL OR OF EVENT FLAGS
	No differences
WTSE$	WAIT FOR SINGLE EVENT FLAG
	No differences

4.2 SYSTEM DIRECTIVE DESCRIPTIONS

Each directive description includes all or most of the following elements, as appropriate:

Name:

The function of the directive in VAX/VMS is described.

Macro Call:

The macro call is shown, each parameter is defined, and the defaults for optional parameters are given in parentheses following the definition of the parameter. Since zero is supplied for most defaulted parameters, only nonzero default values are shown. Parameters ignored by VAX/VMS and RSX-11M are required for compatibility with RSX-11D and IAS.

DSW Return Code:

All return codes that are valid under VAX/VMS are listed and defined. In some cases, a VAX/VMS return status code in parentheses follows an RSX-11M status code. For example:

IE.RSU -- Device allocated to another image (SS$_DEVALLOC)

The VAX/VMS code indicates the VAX/VMS error that caused the corresponding RSX-11M code to be returned.

Some RSX-11M codes reflect several VAX/VMS codes. In this case, VAX/VMS returns the RSX-11M code that it uses by default. Such codes are followed by the phrase "(default error)". For example:

IE.IDU -- Device or unit unknown (default error)

In some cases after a directive failure, VAX/VMS returns an error code that is more meaningful for an I/O operation. In these cases, the high-order byte of the DSW contains 0.

Notes:

The notes presented with some directive descriptions further explain the function, use, and/or consequences of these directives under VAX/VMS. Users should read the notes carefully to ensure proper use of directives.

ABRT$

4.2.1 ABRT$ - ABORT TASK

The ABORT TASK directive instructs the system to terminate the execution of the indicated process's image. The requester can abort itself or an image executing in another process. ABORT TASK is intended for use as an emergency or fault exit.

Macro Call:

```
ABRT$   tsk
```

```
tsk  =  VAX/VMS process name
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Process name unknown (default error)
IE.PRI  --  User not privileged (SS$_NOPRIV)
IE.UPN  --  Insufficient dynamic memory (SS$_INSFMEM)
IE.NOD  --  Image's quota exceeded (SS$_EXQUOTA)
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Force Exit system service to terminate the specified process's image on behalf of the image issuing the ABORT TASK directive.

2. The image issuing the ABORT TASK directive must be executing in a process that meets either of the following requirements:

   - It is in the same group as the process to be aborted and has GROUP privilege.

   - It has WORLD privilege.

3. The exit status is supplied by an exit handling routine (exit handler). It is assumed that the status returns a severe error.

4.2.2 ALTP$ - ALTER PRIORITY

VAX/VMS does not emulate the ALTER PRIORITY directive. Instead, it allows the process for which the directive was issued to continue execution at its present priority.

Macro Call:

```
ALTP$   [tsk][,pri]

  tsk  =  Active task name
  pri  =  New priority, a number from 1 to 250 (decimal)
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB is out of the issuing image's address
            space
IE.SDP  --  DIC or DPB size is invalid
```

ALUN$

4.2.3 ALUN$ - ASSIGN LUN

The ASSIGN LUN directive instructs the system to assign a physical device unit to a logical unit number. An I/O channel is the VAX/VMS equivalent of an RSX-11M logical unit number.

The reassignment of a LUN from one device to another causes VAX/VMS to cancel all I/O requests for the previous assignment.

Macro Call:

```
ALUN$   lun,dev,unt

  lun  =  Logical unit number
  dev  =  Device name (two characters)
  unt  =  Device unit number
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.IDU  --  Device or unit unknown (default error)
IE.ILU  --  Invalid logical unit number
IE.NOD  --  I/O channel quota exceeded (SS$_EXQUOTA)
IE.RSV  --  Device allocated to another image (SS$_DEVALLOC)
IE.DUN  --  Device not mounted (SS$_DEVNOTMOUNT)
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes an Assign I/O Channel system service on behalf of the image issuing the ASSIGN LUN directive.

2. The assignment of RSX-11M device names to VAX/VMS physical devices is described in Section 3.5.

3. If the RSX-11M device name and logical unit number are not assigned as the logical name of a VAX/VMS device, VAX/VMS maps the RSX-11M device name and unit number to an appropriate VAX/VMS device name, controller, and unit number. To perform the mapping, VAX/VMS converts the RSX-11M unit number to decimal and divides it by 16 (decimal). The quotient is added to the ASCII value representing the character A. The result is the controller designation. The remainder becomes the VAX/VMS unit number. The following is an example of the conversion.

   RSX-11M device name and unit number = DB2

   'A'+(2/16) = 'A'+0 with a remainder of 2

   The corresponding VAX/VMS device name, controller letter, and unit number = DBA2.

4. If a LUN is reassigned, its previous assignment is deassigned. The deassignment causes I/O to be canceled on the old assignment. If the attempt to make a new assignment fails, the LUN remains deassigned.

4.2.4 ASTX$S - AST SERVICE EXIT

The AST SERVICE EXIT directive instructs the system to terminate execution of an AST service routine.

If another AST is queued and ASTs are not disabled, VAX/VMS immediately effects the next AST. Otherwise, the system restores the image's state prior to the AST.

Macro Call:

```
ASTX$S   [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.AST  --  Directive not issued from an AST service routine
IE.ADP  --  Part of the DPB or stack is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Note:

1. When an AST occurs, VAX/VMS pushes, at minimum, the following information onto the stack:

```
SP+06,012  --  0
SP+04      --  PS of process prior to AST
SP+02      --  PC of process prior to AST
SP+00      --  DSW of process prior to AST
```

The stack must be in this state when the AST SERVICE EXIT directive is executed.

CLEF$

4.2.5 CLEF$ - CLEAR EVENT FLAG

The CLEAR EVENT FLAG directive instructs the system to clear an indicated event flag and report the flag's polarity before clearing.

Macro Call:

```
CLEF$   efn

  efn  =  Event flag number
```

DSW Return Codes:

```
IS.CLR  --  Successful completion; flag was already clear
IS.SET  --  Successful completion; flag was set
IE.IEF  --  Invalid event flag number because: (1) in the
            range 33 to 64, but no associated common event
            flags; (2) in the range 65 to 96, but no
            associated group global event flags; (3) not in
            the range 1 to 96
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Clear Event Flag system service on behalf of the image issuing the CLEAR EVENT FLAG directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

4.2.6 CMKT$ - CANCEL MARK TIME REQUESTS

The CANCEL MARK TIME REQUESTS directive instructs the system to cancel all mark time requests that were made by the issuing image.

Macro Call:

```
CMKT$   [,,err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Note:

1. VAX/VMS executes a Cancel Timer Request system service specifying that all timer requests be canceled for the image issuing the CANCEL MARK TIME REQUESTS directive.

CRGF$

4.2.7 CRGF$ - CREATE GROUP GLOBAL EVENT FLAGS

The CREATE GROUP GLOBAL EVENT FLAGS directive instructs the system to associate a named common event flag cluster with the process that issued the directive. If the named cluster does not exist, this directive instructs the system first to create the named cluster (with all flags initialized to 0) and then to associate it with the process that issued the directive.

If a CREATE GROUP GLOBAL EVENT FLAGS directive is issued for an event flag cluster that has been marked for deletion (by the ELIMINATE GROUP GLOBAL EVENT FLAGS directive) but has not yet been deleted, the order for deletion is canceled.

Macro Call:

```
CRGF$   [group]

group  =  Group number for the flags to be created
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.UPN  --  Insufficient memory (SS$_INSFMEM)
IE.IUI  --  Invalid group; group must match UIC in task header
            (H.CUIC)
IE.APD  --  Part of the DPB is out of the issuing image's address
            space
IE.DIC  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS issues the Associate Common Event Flag Cluster system service on behalf of the image issuing the CREATE GROUP GLOBAL EVENT FLAGS directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

4. If a group number is specified in the macro call, it must match the group number specified in the task header of the image that issued the call; processes with the same group number then have access to the event flags that are created. If the group number is omitted from the macro call, the group number specified in the task header (H.CUIC) is used.

4.2.8 CSRQ$ - CANCEL TIME BASED INITIATION REQUESTS

The CANCEL TIME BASED INITIATION REQUESTS directive instructs the system to cancel all time-synchronized wake requests for a specified process's image regardless of the source of each request.

Macro Call:

```
CSRQ$   tsk
```

tsk = VAX/VMS process name

DSW Return Codes:

IS.SUC	--	Successful completion
IE.INS	--	Specified process name unknown (default error)
IE.PRI	--	Privilege violation (SS$_NOPRIV)
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Cancel Wakeup system service on behalf of the image issuing the CANCEL TIME BASED INITIATION REQUESTS directive.

2. The image issuing the CANCEL TIME BASED INITIATION REQUESTS directive must be executing in a process that meets either of the following requirements:

   - It is in the same group as the process for which requests are to be canceled and has GROUP privilege.

   - It has WORLD privilege.

DECL$S

4.2.9 DECL$S - DECLARE SIGNIFICANT EVENT

The DECLARE SIGNIFICANT EVENT directive instructs the system to declare a significant event. Declaration of a significant event has no effect on a VAX/VMS system; the concept of a significant event in the RSX-11M sense does not exist under VAX/VMS.

Macro Call:

```
DECL$S  [,err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Note:

1. No operation is performed and a success status is returned.

DSAR$S or IHAR$S

4.2.10 DSAR$S (or IHAR$S) - DISABLE (or INHIBIT) AST RECOGNITION

The DISABLE AST RECOGNITION directive instructs the system to disable recognition of user-level ASTs for the issuing image. The ASTs are queued as they occur and are effected when the image enables AST recognition. When an AST service routine is executing, AST recognition also is disabled. The initial state of an image is to have recognition enabled.

Macro Call:

```
DSAR$S  [err]
  or
IHAR$S  [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ITS  --  AST recognition is already disabled
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Note:

1. While disabled, ASTs are queued in a first-in/first-out list.

DSCP$S

4.2.11 DSCP$S - DISABLE CHECKPOINTING

The DISABLE CHECKPOINTING directive instructs the system to disable swapping for the process.

Macro Call:

```
DSCP$S  [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ITS  --  Swapping already disabled
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Set Swap Mode system service on behalf of the image issuing the DISABLE CHECKPOINTING directive.

2. The image's initial state has swapping enabled.

3. The requesting image must have the privilege to set its swap mode (PSWAPM).

4.2.12 ELGF$ - ELIMINATE GROUP GLOBAL EVENT FLAGS

The ELIMINATE GROUP GLOBAL EVENT FLAGS directive instructs the system to disassociate the calling process from a named common event flag cluster.

If no other processes are associated with a cluster thus marked for deletion, the cluster is deleted immediately. If, however, the cluster is still associated with other processes, it is not deleted until all of these processes are disassociated from the cluster.

Macro Call:

```
ELGF$   [group]

group  =  Group number of flags to be eliminated
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.IUI  --  Invalid group; group must match UIC in task header
            (H.CUIC)
IE.ADP  --  Part of the DPB is out of the issuing image's address
            space
IE.DIC  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS issues the Disassociate Common Event Flag Cluster system service on behalf of the image issuing the ELIMINATE GROUP GLOBAL EVENT FLAGS directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

4. If a group number is specified in the macro call, it must match the group number specified in the task header of the image that issued the call. If the group number is omitted from the macro call, the group number specified in the task header (H.CUIC) is used.

ENAR$S

4.2.13 ENAR$S - ENABLE AST RECOGNITION

The ENABLE AST RECOGNITION directive instructs the system to recognize user-level ASTs for the issuing image; that is, the directive nullifies a DISABLE AST RECOGNITION directive. ASTs that were queued while recognition was disabled are effected when the ENABLE AST RECOGNITION directive is issued. The initial state of an image is to have AST recognition enabled.

Macro Call:

```
ENAR$S   [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ITS  --  AST recognition is not disabled
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

4.2.14 ENCP$S - ENABLE CHECKPOINTING

The ENABLE CHECKPOINTING directive instructs the system to enable swapping for the process.

Macro Call:

```
ENCP$S   [err ]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ITS  --  Swapping already enabled
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Set Swap Mode system service on behalf of the image issuing the ENABLE CHECKPOINTING directive.

2. The image's initial state has swapping enabled.

3. The requesting image's process must have the PSWAPM privilege to set its swap mode.

EXIF$

4.2.15 EXIF$ - EXIT IF

The EXIT IF directive instructs the system to terminate execution of the issuing image if the specified event flag is not set. VAX/VMS returns control to the issuing image if the specified event flag is set.

Macro Call:

```
EXIF$   efn

  efn  =  Event flag number
```

DSW Return Codes:

IS.SET	--	Indicated event flag is set; image did not exit
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

2. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

3. This service is not interlocked in VAX/VMS; it is interlocked in RSX-11M. That is, under VAX/VMS, the conditional exit if function is accomplished using two system service calls: Read Event Flags and, if necessary, Exit.

4.2.16 EXIT$S - TASK EXIT

The TASK EXIT directive instructs the system to terminate execution of the issuing image.

Macro Call:

```
EXIT$S  [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. A return to the image occurs only if the directive is rejected.

2. VAX/VMS executes an Exit system service on behalf of the issuing image. The success status is returned.

EXST$

4.2.17 EXST$ - EXIT WITH STATUS

The EXIT WITH STATUS directive instructs the system to terminate execution of the issuing image and to accept from the image a status code indicating whether the termination is normal or abnormal.

Macro Call:

```
EXST$   sts[,err]

  sts  =  exit status

          EX$SUC -- Normal termination (RSX$_EXITSTATUS)
          EX$WAR -- Warning (RSX$_EXITSTATUS)
          EX$ERR -- Abnormal termination (RSX$_EXITSTATUS)
          EX$SEV -- Severe error termination (RSX$_EXITSTATUS)

  err  =  Error routine address
```

DSW Return Codes:

```
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. A return to the image occurs only if the directive is rejected.

2. VAX/VMS executes an Exit system service specifying the exit status of the image.

4.2.18 EXTK$ - EXTEND TASK

The EXTEND TASK directive instructs the system to modify the size of the issuing task by a positive or negative increment of 32-word blocks. If the directive does not specify an increment value, VAX/VMS makes the issuing image's size equal to its initial size.

Macro Call:

```
EXTK$   [inc]
```

inc = A positive or negative number equal to the number of 32-word blocks by which the image size is to be extended or reduced

DSW Return Codes:

IS.SUC	--	Successful completion
IE.ALG	--	The issuing image attempted to reduce its size to less than the size of its header; or the image tried to increase its size beyond 32K words or beyond the base of the lowest mapped library or common block
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. An image cannot extend itself past its 64K byte (32K words) address space or, if libraries or common areas are present, past the base of the lowest mapped library or common block.

2. An image can extend itself to the base of its read-only section.

GLUN$

4.2.19 GLUN$ - GET LUN INFORMATION

The GET LUN INFORMATION directive instructs the system to fill a 6-word buffer with information about a physical device unit to which a LUN is assigned.

Macro Call:

```
GLUN$   lun,buf
```

lun = Logical unit number
buf = Address of 6-word buffer that is to receive the LUN information

Buffer Format:

WD. 00 Name of assigned device

WD. 01 Unit number of assigned device in the low-order byte. The high-order bit of the word is set to indicate that the driver is loaded.

WD. 02 First device characteristics word:

```
Bit 0  --  Record-oriented device (1=yes) [FD.REC][1]
Bit 1  --  Carriage-control device (1=yes)[FD.CCL]
Bit 2  --  Terminal device (1=yes)[FD.TTY]
Bit 3  --  Directory device (1=yes)[FD.DIR]
Bit 4  --  Single directory device (1=yes)[FD.SDI]
Bit 5  --  Sequential device (1=yes)[FD.SQD]
Bit 6  --  Spooled device
Bit 7  --  Reserved
Bit 8  --  Reserved
Bit 9  --  Reserved
Bit 10 --  User-mode diagnostics supported
Bit 11 --  Unit software write locked (1=yes)
Bit 12 --  0
Bit 13 --  0
Bit 14 --  Device mountable as a Files-11 device
           (1=yes)
Bit 15 --  Device mountable (1=yes)
```

WD. 03, 04 VAX/VMS device-dependent longword. The contents of this longword are described in the VAX/VMS I/O User's Guide. For a disk, the longword contains the maximum number of blocks; this number is compatible with RSX-11M.

WD. 05 Standard device buffer size

---

1. Bits with associated symbols have the symbols shown in square brackets. These symbols can be defined for use by an image by means of the FCSBT$ macro. See the IAS/RSX-11 I/O Operations Reference Manual.

DSW Return Codes:

IS.SUC	--	Successful completion
IE.ULN	--	Unassigned LUN
IE.ILU	--	Invalid logical unit number
IE.ADP	--	Part of the DPB or buffer is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Get Channel Information system service on behalf of the image issuing the GET LUN INFORMATION directive.

2. VAX/VMS converts the name and unit number of the VAX/VMS device to which the LUN is assigned to an RSX-11M device name and unit number before returning the LUN information.

   To convert from a VAX/VMS device name to the RSX-11M form, VAX/VMS subtracts the value representing the ASCII character A (65) from the value of the ASCII character representing the controller letter and multiplies the result by 16 (decimal). It then adds the VAX/VMS unit number, converted from decimal to octal. The final result is an RSX-11M unit number that is appended to the 2-character device name. For example, the VAX/VMS device name TTA2 converts to the RSX-11M device name TT2.

   TTA2 to TT2:

   ```
   Unit = ('A'-'A')*16+2 = 0*16+2 = 2
   ```

3. If the device to which the LUN is assigned is a spooled device (for example, a line printer), VAX/VMS returns the characteristics of the intermediate device (for example, disk).

4. Mailboxes have 16-bit unit numbers. The low-order 8 bits are returned by GET LUN INFORMATION in word 1. Mailboxes must be referred to using a logical name rather than using the unit number returned.

GMCR$

4.2.20 GMCR$ - GET MCR COMMAND LINE

The GET MCR COMMAND LINE directive instructs the system to transfer an 80-byte command line to the issuing image. It is the command line used to invoke the image. As a result, it can be in either MCR or DCL format.

Macro Call:

```
GMCR$
```

DSW Return Codes:

+n	--	Successful completion; n is the number of data bytes transferred, excluding the termination character. The termination character is, however, in the buffer
IE.AST	--	No command line exists for the issuing image; that is, the image was not requested by a command other than RUN or the image has already issued the GET MCR COMMAND LINE directive
IE.ADP	--	Part of the DPB is out of the issuing process's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. The system processes all lines to:
   - Convert tabs to a single space
   - Convert multiple spaces to a single space
   - Convert lowercase characters to uppercase
   - Remove all trailing blanks
2. The terminator <CR> is the last character in the line.
3. The command line can be the result of the following types of user-issued DCL commands:

Format	Example
MCR name command-string	$ MCR PIP LP:=MYFILE
MCR MCR> name command-string	$ MCR MCR>PIP LP:=MYFILE

4. The command line can be the result of he following types of MCR commands:

Format	Example
>name command-string	>PIP LP:=MYFILE
>name followed by prompt	>PIP PIP>

5. The command line received as a result of the GET MCR COMMAND LINE directive varies depending on the format of the command typed. If the command contains a command string, for example, LP:=MYFILE, the command containing that string and its length are available to the image. Note that if the command was issued from DCL, the command name and any blanks following the command name are removed before making it available to the image.

   Example:

   ```
   $ MCR PIP LP:=MYFILE
   > PIP LP:=MYFILE
   ```

   Both of these commands result in the same string, and its length of 14, being available to the image using the GMCR$ directive, where the actual string is:

   ```
   PIP LP:=MYFILE
   ```

   If no string is supplied, VAX/VMS returns a command string length of 0.

6. When an image executes as a result of a RUN command (either DCL or MCR), the command line length is 0.

GPRT$

4.2.21 GPRT$ - GET PARTITION PARAMETERS

The GET PARTITION PARAMETERS directive instructs the system to fill an indicated 3-word buffer with partition parameters. The VAX/VMS system does not have the concept of partitions. Therefore, the data returned is consistent with that returned by RSX-11M for a system-controlled partition named GEN starting at 40000(8).

Macro Call:

```
GPRT$   [prt],buf
```

```
prt  =  Partition name
buf  =  Address of a 3-word buffer
```

The buffer has the following format:

```
WD. 0  --  400(8)

WD. 1  --  Image size, including all libraries, expressed as
           a multiple of 64 bytes

WD. 2  --  A value of 0 is returned to indicate a
           system-controlled partition
```

DSW Return Codes:

Successful completion is indicated by carry clear and $DSW equal to 0 indicating a mapped system.

```
IE.ADP  --  Part of the DPB or buffer is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

4.2.22 GTIM$ - GET TIME PARAMETERS

The GET TIME PARAMETERS directive instructs the system to fill an indicated 8-word buffer with the current time parameters. All time parameters are delivered as binary numbers. The value ranges are shown in decimal below.

Macro Call:

```
GTIM$   buf

  buf  =  Address of 8-word buffer
```

The buffer has the following format:

```
WD. 0  --  Year (since 1900)
WD. 1  --  Month (1-12)
WD. 2  --  Day (1-31)
WD. 3  --  Hour (0-23)
WD. 4  --  Minute (0-59)
WD. 5  --  Second (0-59)
WD. 6  --  Tick of second (0-99)
WD. 7  --  Ticks per second (100 ticks occur per second)
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB or buffer is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Get Time system service for the image issuing the GET TIME PARAMETERS directive.

2. VAX/VMS provides a 100-tick-per-second clock.

GTSK$

4.2.23 GTSK$ - GET TASK PARAMETERS

The GET TASK PARAMETERS directive instructs the system to fill an indicated 16-word buffer with parameters relating to the issuing process.

Macro Call:

```
GTSK$   buf

  buf  =  Address of a 16-word buffer
```

The buffer has the following format:

```
WD. 00  --  Process name in Radix-50 (first half), if any
WD. 01  --  Process name in Radix-50 (second half), if any
WD. 02  --  Partition name in Radix-50 (GEN)
WD. 03  --  Partition name in Radix-50 (blanks)
WD. 04  --  Undefined
WD. 05  --  Undefined
WD. 06  --  Run priority from RSX-11M task header
WD. 07  --  Low-order bytes of the UIC group and number codes
WD. 08  --  Number of logical I/O units (LUNs)
WD. 09  --  Undefined
WD. 10  --  Undefined
WD. 11  --  Address of task SST vector tables
WD. 12  --  Size of task SST vector table in words
WD. 13  --  Size in bytes of image's address window excluding
            libraries and common areas
WD. 14  --  System in which process is running; 5 for VAX/VMS
WD. 15  --  High-order bytes of the UIC group and member codes
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB or buffer is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB is invalid
```

4.2.24 MRKT$ - MARK TIME

The MARK TIME directive instructs the system to set an event flag and/or declare an AST after an indicated time interval. The interval begins when the image issues the directive. If an event flag is specified, the flag is cleared when the directive is issued and set when the interval elapses. If an AST entry point address is specified, an AST occurs when the interval elapses.

Macro Call:

```
MRKT$   [efn],tmg,tnt[,ast]

  efn  =  Event flag number
  tmg  =  Time interval magnitude
  tnt  =  Time interval unit
  ast  =  AST entry point address
```

DSW Return Codes:

IS.SUC	--	Successful completion
IE.ITI	--	Invalid time parameter
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.UPN	--	Insufficient dynamic memory (SS$_INSFMEM)
IE.NOD	--	Image's quota exceeded (SS$_EXQUOTA)
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Set Timer system service on behalf of the process issuing the MARK TIME directive.

2. If an AST entry point address is specified, the AST service routine is entered with the stack in the following state:

```
SP+08,14 - 0
SP+06    - PS of process prior to AST
SP+04    - PC of process prior to AST
SP+02    - DSW of process prior to AST
SP+00    - Event flag number or 0 (if none was
           specified in the MARK TIME directive)
```

   The event flag number must be removed from the stack before an AST SERVICE EXIT directive is executed.

3. VAX/VMS returns the DSW code IE.ITI if the directive specifies an invalid time parameter. The time parameter consists of two components: the time interval magnitude (tmg) and the time interval unit (tnt).

A legal magnitude value (tmg) is related to the value assigned to the time interval unit (tnt). The unit values are encoded as follows:

1 = Ticks (1/100 of a second per tick)
2 = Seconds
3 = Minutes
4 = Hours

The magnitude (tmg) is the number of units to be clocked. The following list describes the magnitude values that are valid for each type of unit. In no case can the value of tmg exceed 24 hours.

If tnt = 0, 1, or 2, tmg can be any positive value with a maximum of 15 bits.

If tnt = 3, tmg can have a maximum value of 1440(10).

If tnt = 4, tmg can have a maximum value of 24(10).

4. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

5. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

6. VAX/VMS enforces a quota on the number of ASTs that a process can have pending.

4.2.25 QIO$ - QUEUE I/O REQUEST

The QUEUE I/O REQUEST directive instructs the system to place an I/O request for an indicated physical device unit into a queue of priority-ordered requests for that device unit. The physical device unit is specified as a logical unit number (LUN).

If the directive call specifies an event flag, VAX/VMS clears the flag when the request is queued and sets the flag upon request completion.

The I/O status block is also cleared when the request is queued, and set to the final I/O status when the I/O request is complete. If an AST service routine entry point address is specified, the AST occurs upon I/O completion, and the process's WAITFOR mask word, PS, PC, DSW (directive status), and the address of the I/O status block are pushed onto the stack.

Macro Call:

```
QIO$    fnc,lun,[efn],[pri],[isb],[ast][,prl]

  fnc  =  I/O function code
  lun  =  Logical unit number
  efn  =  Event flag number
  pri  =  Priority; ignored, but must be present
  isb  =  Address of I/O status block
  ast  =  Address of AST service routine entry point
  prl  =  Parameter list of the form <P1,...,P6>
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ULN  --  Unassigned LUN
IE.ILU  --  Invalid LUN
IE.IEF  --  Invalid event flag number because:  (1) in the
            range 33 to 64, but no associated common event
            flags; (2) in the range 65 to 96, but no
            associated group global event flags; (3) not in
            the range 1 to 96
IE.NOD  --  Quota exceeded (SS$_EXQUOTA)
IE.UPN  --  Insufficient memory (SS$_INSFMEM)
IE.ADP  --  Part of the DPB or I/O status block is out of the
            issuing image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Queue I/O Request system service on behalf of the image issuing the QUEUE I/O REQUEST directive.

2. Chapter 5 explains function codes, parameter meanings, and I/O status block return values.

3. If the directive call specifies an AST entry point address, the process enters the AST service routine with the stack in the following state:

   SP+16 - SP+10 - 0
   SP+06 - PS of process prior to AST
   SP+04 - PC of process prior to AST
   SP+02 - DSW of process prior to AST
   SP+00 - Address of I/O status block, or 0 if none was specified in the QIO directive.

   The address of the I/O status block, which is a trap-dependent parameter, must be removed from the stack before an AST SERVICE EXIT directive is executed.

4. VAX/VMS pushes four words of zeros in SP+16 through SP+10. RSX-11M pushes three words with undefined contents and a 1-word event flag mask.

5. If the directive is rejected, the specified event flag is not guaranteed to be cleared or set. Therefore a process that waits for a rejected QUEUE I/O REQUEST AND WAIT directive may wait forever.

6. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

7. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

8. VAX/VMS enforces a quota on the number of ASTs that a process can have pending.

4.2.26 QIOW$ - QUEUE I/O REQUEST AND WAIT

The QUEUE I/O REQUEST AND WAIT directive is identical to QUEUE I/O REQUEST with one exception: if the wait variation of the directive specifies an event flag, VAX/VMS automatically effects a WAIT FOR SINGLE EVENT FLAG directive. If an event flag is not specified, however, VAX/VMS treats the directive as if it were a QUEUE I/O REQUEST.

Macro Call:

```
QIOW$   fnc,lun,efn,[pri],[isb],[ast][,prl]

  fnc  =  I/O function code
  lun  =  Logical unit number
  efn  =  Event flag number
  pri  =  Priority; ignored, but must be present
  isb  =  Address of I/O status block
  ast  =  Address of AST service routine entry point
  prl  =  Parameter list of the form <P1,...,P6>
```

DSW Return Codes:

IS.SUC	--	Successful completion
IE.ULN	--	Unassigned LUN
IE.ILU	--	Invalid LUN
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.UPN	--	Insufficient memory (SS$_INSFMEM)
IE.NOD	--	Quota exceeded (SS$_EXQUOTA)
IE.ADP	--	Part of the DPB or I/O status block is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Queue I/O Request and Wait for Event Flag system service on behalf of the image issuing the QUEUE I/O REQUEST AND WAIT directive.

2. See the Notes for the QUEUE I/O REQUEST directive.

RCST$

4.2.27 RCST$ - RECEIVE DATA OR STOP

The RECEIVE DATA OR STOP directive instructs the system to attempt to read a message from the mailbox created when the image containing the directive was loaded. If no data has been sent to the mailbox by another process, the process that issued the directive is stopped, or placed in hibernation. This directive cannot be issued from an AST service routine.

A 2-word sending process name in Radix-50 form and the 13-word data block are returned in a 15-word buffer.

Macro Call:

```
RCST$   [tname],buf
```

```
tname  =  Sending task name; ignored under VAX/VMS
  buf  =  Address of 15-word buffer
```

DSW Return Codes:

IS.SUC	--	Successful completion; data read from mailbox
IE.NOD	--	Quota exceeded (SS$_EXQUOTA)
IE.UPN	--	Insufficient memory (SS$_INSFMEM)
IE.SET	--	No data obtained from mailbox or no mailbox exists; process stopped
IE.AST	--	Directive was issued from an AST service routine and no data obtained from mailbox or no mailbox exists
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Queue I/O Request system service and, if appropriate, a Hibernate system service on behalf of the process issuing the RECEIVE DATA OR STOP directive. The I/O operation reads data from a mailbox associated with the process by VAX/VMS when it loaded the image.

2. The name of the mailbox is RCVD followed by the process name, that is, RCVDname.

3. The mailbox is not created until the image actually begins to execute.

4. The image issuing the receive directive must have a name specified at task-build time; that is, the image label block must contain a task name. VAX/VMS uses the presence of the task name as a indication that the image may receive data and sets up the necessary machanism.

5. Because the protection of the mailbox is specified as read access for the owner (receiving process) and write access for the group (sending processes), this directive is useful only for passing data between processes within the same group.

6. If no data is obtained from the mailbox or if no mailbox exists, VAX/VMS executes a Hibernate system service for the process that issued the RECEIVE DATA OR STOP directive, and a status code of IS.SET is returned. Note that the status code IS.SET cannot be seen by the process that issues the directive until the process is restarted by an UNSTOP TASK directive (a Wake system service).

7. If a process issued the RECEIVE DATA OR STOP directive and stops because no data is available in the mailbox, the process is not automatically awakened later when data is placed into the mailbox. A task sending data should also wake the stopped task using the UNSTOP TASK directive; to receive the data, the awakened task must issue another receive-data directive (RECEIVE DATA, RECEIVE DATA OR EXIT, or RECEIVE DATA OR STOP).

RCVD$

4.2.28 RCVD$ - RECEIVE DATA

The RECEIVE DATA directive instructs the system to read a message from the mailbox created when the image was loaded and place that message in a buffer within the issuing image.

A 2-word sending process name in Radix-50 form and the 13-word data block are returned in a 15-word buffer.

Macro Call:

```
RCVD$ [tsk],buf
```

```
buf = Address of 15-word buffer
tsk = Sending task name;  ignored under VAX/VMS
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.NOD  --  Quota exceeded (SS$_EXQUOTA)
IE.UPN  --  Insufficient memory (SS$_INSFMEM)
IE.ITS  --  No data currently available in mailbox or no
            mailbox (default error)
IE.ADP  --  Part of the DPB or buffer is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a read Queue I/O Request system service on behalf of the process issuing the RECEIVE DATA directive. The I/O operation reads data from a mailbox associated with the process by VAX/VMS when it loads the image.

2. The name of the mailbox is RCVD followed by the process name, that is, RCVDname.

3. The mailbox is not created until the image actually begins to execute.

4. Because protection is specified as read access for the owner (receiving process) and write access for the group (sending processes), this directive is useful only for passing data between processes within the same group.

5. The image issuing the receive directives must have a name specified at task-build time; that is, the image label block must contain a task name. VAX/VMS uses the presence of the task name as an indication that the image may receive data and sets up the necessary mechanism.

4.2.29 RCVX$ - RECEIVE DATA OR EXIT

The RECEIVE DATA OR EXIT directive instructs the system to read a message from the mailbox created when the image was loaded. The message was sent to the mailbox previously by another process. If no data has been sent, the image exits.

A 2-word sending process name in Radix-50 form and the 13-word data block are returned in a 15-word buffer.

Macro Call:

```
RCVX$   [tsk],buf

  buf  =  Address of 15-word buffer
  tsk  =  Sending task name; ignored under VAX/VMS
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.NOD  --  Quota exceeded (SS$_EXQUOTA)
IE.UPN  --  Insufficient memory (SS$_INSFMEM)
IE.ADP  --  Part of the DPB or buffer is out of the  issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Queue I/O Request system service and, if appropriate, an Exit system service on behalf of the process issuing the RECEIVE DATA OR EXIT directive. The I/O operation reads data from a mailbox associated with the process by VAX/VMS when it loaded the image.

2. The name of the mailbox is RCVD followed by the process name, that is, RCVDname.

3. The mailbox is not created until the image actually begins to execute.

4. Because protection is specified as read access for the owner (receiving process) and write access for the group (sending processes), this directive is useful only for passing data between processes within the same group.

5. If no data is obtained from the mailbox, VAX/VMS executes an Exit system service for the image. The exit status is SS$_NORMAL.

6. The image issuing the receive directives must have a name specified at task-build time; that is, the image label block must contain a task name. VAX/VMS uses the presence of the task name as an indication that the image may receive data and sets up the necessary mechanism.

7. This directive does not provide the same interlock between the sender and the receiver as it does in RSX-11M.

8. If no mailbox exists, the image exits with a success status.

RDAF$

4.2.30 RDAF$ - READ ALL EVENT FLAGS

The READ ALL EVENT FLAGS directive instructs the system to read local and common event flags for the issuing process and record their values in a 64-bit (4-word) buffer.

Macro Call:

```
RDAF$   buf
```

The buffer has the following format:

```
WD. 00  --  Local flags 1 through 16
WD. 01  --  Local flags 17 through 32
WD. 02  --  Common flags 33 through 48
WD. 03  --  Common flags 49 through 64
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB or buffer is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS issues a Read Event Flags system service on behalf of the image issuing the READ ALL EVENT FLAGS directive.

2. The READ ALL EVENT FLAGS directive does not read group global flags. The READ EXTENDED EVENT FLAGS directive reads all 96 flags.

3. If no common event flag cluster is associated with the process, the common event flags are returned as all zeros.

4. A task image must be associated with common flags to access flags in the common cluster (see Section 2.5).

5. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95

4.2.31 RDXF$ - READ EXTENDED EVENT FLAGS

The READ EXTENDED EVENT FLAGS directive instructs the system to read all local, common, and group global event flags for the issuing process and to record their values in a 96-bit (6-word) buffer.

Macro Call:

```
RDXF$   buf

  buf  =  Address of 6-word buffer
```

The buffer has the following format:

```
WD. 00  =  Local flags 1 through 16
WD. 01  =  Local flags 17 through 32
WD. 02  =  Common flags 33 through 48
WD. 03  =  Common flags 49 through 64
WD. 04  =  Group global flags 65 through 80
WD. 05  =  Group global flags 81 through 96
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IS.CLR  --  Group global event flags do not exist.  Words 4 and 5
            of the buffer contain zeros
IE.ADP  --  Part of the DPB or buffer is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS issues the Read Event Flags system service on behalf of the image issuing the READ EXTENDED EVENT FLAGS directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

4. If no common event flag cluster is associated with the process, the common event flags are returned as all zeros.

5. If no group global event flag cluster is associated with the process, the group global event flags are returned as all zeros and IS.CLR is returned.

RQST$

4.2.32 RQST$ - REQUEST TASK

The REQUEST TASK directive instructs the system to activate a hibernating process. Chapter 2 describes the use of the Hibernate and Wake system services for real-time images.

REQUEST TASK is a frequently used subset of the RUN directive.

Macro Call:

```
RQST$   tsk,[prt],[pri][,ugc,umc]

  tsk  =  VAX/VMS process name
  prt  =  Partition name; ignored
  pri  =  Priority; ignored
  ugc  =  UIC group code; ignored
  umc  =  UIC member code; ignored
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Process name not known (default error)
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.UPN  --  Insufficient dynamic memory (SS$_INSFMEM)
IE.NOD  --  Process quota exceeded (SS$_EXQUOTA)
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Wake system service on behalf of the process issuing the REQUEST TASK directive.

2. The requested process must currently be present in the system; that is, either hibernating or active.

3. The image issuing the REQUEST TASK directive must be executing in a process that meets either of the following requirements:

   - It is in the same group as the requested process and has GROUP privilege.

   - It has WORLD privilege.

4. VAX/VMS maintains an indicator to determine whether any wake requests have been issued for an active process. If the pending wake indicator is set and the process issues a hibernate request, the process remains active, and the pending wake indicator is cleared. A subsequent hibernate request causes the process to hibernate.

5. Hibernations that are caused by STOP directives (RECEIVE DATA OR STOP, STOP, STOP FOR LOGICAL OR OF EVENT FLAGS, and STOP FOR SINGLE EVENT FLAG) cannot be reactivated by the REQUEST TASK directive.

4.2.33 RSUM$ - RESUME TASK

The RESUME TASK directive instructs the system to awaken a process that is in a state of hibernation.

Macro Call:

```
RSUM$   tsk
```

tsk = VAX/VMS process name

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Process name unknown (default error)
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.UPN  --  Insufficient dynamic memory (SS$_INSFMEM)
IE.NOD  --  Image's quota exceeded (SS$_EXQUOTA)
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Wake system service on behalf of the process issuing the RESUME TASK directive.

2. The image issuing the RESUME TASK directive must be executing in a process that meets either of the following requirements:

   - The process is in the same group as the process to be resumed and has GROUP privilege.

   - The process has WORLD privilege.

3. VAX/VMS maintains an indicator to determine whether any wake requests have been issued for an active process. If the indicator is set and the process issues a hibernate request, the process remains active, and the indicator is cleared. A subsequent hibernate (SUSPEND) request causes the process to hibernate.

4. If a RESUME TASK directive is issued for an image that is active, the status returned is success. The process remains active.

5. Hibernations that are caused by STOP directives (RECEIVE DATA OR STOP, STOP, STOP FOR LOGICAL OR OF EVENT FLAGS, and STOP FOR SINGLE EVENT FLAG) cannot be reactivated by the RESUME TASK directive.

RUN$

4.2.34 RUN$ - RUN TASK

The RUN TASK directive requests the system to activate a hibernating process at a specified future time and, optionally, to reactivate that process periodically. The schedule time is specified in terms of delta time from issuance. If the smg, rmg, and rnt parameters are omitted, RUN TASK is the same as REQUEST TASK except that RUN TASK causes the process to become active one clock tick after the directive is issued.

Macro Call:

```
RUN$    tsk,[prt],[pri],[ugc],[umc],[smg],snt[,rmg,rnt]
```

```
tsk  =  VAX/VMS process name
prt  =  Partition name; ignored
pri  =  Priority; ignored
ugc  =  UIC group code; ignored
umc  =  UIC member code; ignored
smg  =  Schedule delta magnitude
snt  =  Schedule delta unit
rmg  =  Reschedule interval magnitude
rnt  =  Reschedule interval unit
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Process name unknown
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.UPN  --  Insufficient dynamic memory (SS$_INSFMEM)
IE.NOD  --  Process quota exceeded (SS$_EXQUOTA)
IE.ITI  --  Invalid time parameter
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Schedule Wake-up system service on behalf of the process issuing the RUN TASK directive.

2. The target process must be present in the system.

3. The image issuing the RUN TASK directive must be executing in a process that meets either of the following requirements:

   - It is in the same group as the process to be run and has GROUP privilege.

   - It has WORLD privilege.

4. VAX/VMS maintains an indicator to determine whether any wake requests have been issued for an active process. If the wake pending indicator is set and the process issues a hibernate request, the process remains active, and the wake pending indicator is cleared. A subsequent hibernate (SUSPEND) request causes the process to hibernate.

5. VAX/VMS returns the DSW code IE.ITI if the directive specifies an invalid time parameter. A time parameter consists of two components: the time interval magnitude (smg or rmg) and the time interval unit (snt or rnt).

   A legal magnitude value (smg or rmg) is related to the value assigned to the time interval unit snt or rnt. The unit values are encoded as follows:

   1 = Ticks (1/100 of a second per tick)
   2 = Seconds
   3 = Minutes
   4 = Hours

   The magnitude is the number of units to be clocked. The following list describes the magnitude values that are valid for each type of unit. In no case can the magnitude exceed 24 hours.

   - If unit equals 0, 1, or 2, the magnitude can be any positive value with a maximum of 15 bits.
   - If unit equals 3, the magnitude can have a maximum value of 1440(10).
   - If unit equals 4, the magnitude can have a maximum value of 24(10).

6. The schedule delta time is the difference in time from the issuance of the RUN TASK directive to the time the process is to be run. This time can be specified in the range from one-clock-tick to 24 hours.

7. Hibernations that are caused by STOP directives (RECEIVE DATA OR STOP, STOP, STOP FOR LOGICAL OR OF EVENT FLAGS, and STOP FOR SINGLE EVENT FLAG) cannot be reactivated by the RUN TASK directive.

SDAT$

4.2.35 SDAT$ - SEND DATA

The SEND DATA directive instructs the system to send a 13-word message to a mailbox.

When an event flag is specified in the SEND DATA directive, the indicated flag is set for the sending process.

Macro Call:

```
SDAT$   tsk,buf[,efn]

  tsk  =  VAX/VMS process name
  buf  =  Address of 13-word data buffer
  efn  =  Event flag number
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Receiver process name unknown (default error)
IE.NOD  --  Quota exceeded (SS$_EXQUOTA)
IE.UPN  --  Insufficient memory (SS$_INSFMEM)
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.IEF  --  Invalid event flag number because:  (1) in the
            range 33 to 64, but no associated common event
            flags; (2) in the range 65 to 96, but no
            associated group global event flags; (3) not in
            the range 1 to 96
IE.ADP  --  Part of DPB or data block is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a write Queue I/O Request system service on behalf of the process issuing the SEND DATA directive. The I/O operation writes to a mailbox named RCVD followed by the specified process name, that is, RCVDname.

2. The sending process must be in the same group as the receiving process because protection allows the group write access to the mailbox and denies access to the world.

3. The target process must be executing an image that had a name specified at task-build time; that is, the image label block must contain a task name. VAX/VMS uses the presence of the task name as an indication that the image may receive data and sets up the necessary mechanism.

4. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

5. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

SETF$

4.2.36 SETF$ - SET EVENT FLAG

The SET EVENT FLAG directive instructs the system to set an indicated event flag and report the flag's previous value.

Macro Call:

```
SETF$   efn

  efn  =  Event flag number
```

DSW Return Codes:

IS.CLR	--	Flag was clear
IS.SET	--	Flag was already set
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Set Event Flag system service on behalf of the image issuing the SET EVENT FLAG directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

4.2.37 SFPA$ - SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST

The SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST directive instructs the system either to enable or disable delivery of floating-point processor exception ASTs.

When an AST service routine entry point address is specified, future floating-point processor exception ASTs occur for the issuing process, and control is transferred to the indicated location at the time of the AST's occurrence. When an AST service entry point address is not specified, future floating-point processor exception ASTs do not occur until the image issues a directive that specifies an AST entry point.

Macro Call:

```
SFPA$   [ast]

  ast  =  AST service routine entry point address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.UPN  --  Insufficient dynamic memory (SS$_INSFMEM)
IE.ITS  --  AST entry point address is already unspecified
IE.AST  --  Directive was issued from an AST service routine
            or ASTs are disabled
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. The SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST requires dynamic memory.

2. VAX/VMS queues floating-point processor exception ASTs when a floating-point processor exception trap occurs for the task. No future floating-point processor exception ASTs are queued for the process until the first one queued has actually been effected.

3. The floating-point processor exception AST service routine is entered with the task stack in the following state:

   ```
   SP+12 - Event flag mask word
   SP+10 - PS of task prior to AST
   SP+06 - PC of task prior to AST
   SP+04 - DSW of task prior to AST
   SP+02 - Floating exception code
   SP+00 - Floating exception address
   ```

   The image must remove the floating-point exception code and address
   from the stack before an AST SERVICE EXIT directive is executed.

4. This directive cannot be issued from an AST service routine or when ASTs are disabled.

SPND$S

4.2.38 SPND$S - SUSPEND

The SUSPEND directive instructs the system to place the process in a state of hibernation. An image can suspend only the process in which it is executing. The suspended process can be restarted by another process that issues a RESUME directive for it.

Macro Call:

```
SPND$S  [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Process has no name
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Hibernate system service on behalf of the process issuing the SUSPEND directive.

2. A suspended process retains control of the system resources allocated to it. VAX/VMS makes no attempt to free these resources.

3. VAX/VMS maintains an indicator to determine whether any wake requests have been issued for an active process. If the indicator is set and the process issues a hibernate request, the process remains active, and the indicator is cleared. A subsequent hibernate request causes the process to hibernate.

4. If a SUSPEND directive is issued by an image that has pending resume requests, the following occurs.

   The status returned is success.

   The process remains active.

   The wake-pending indicator is cleared.

5. A process can be resumed only by specifying its process name; therefore, a process is not allowed to suspend itself unless it has a process name.

6. A process that is hibernating because of a SUSPEND directive cannot be reactivated by the UNSTOP TASK directive.

4.2.39 SPRA$ - SPECIFY POWER RECOVERY AST

The SPECIFY POWER RECOVERY AST directive instructs the system to record either of the following:

- That power-recovery ASTs for the issuing process are required, and the address to which to transfer control when a powerfail recovery AST occurs
- That power-recovery ASTs for the issuing process are no longer required

When an AST service routine entry point address is specified, future power-recovery ASTs occur for the issuing process. VAX/VMS transfers control to the specified address whenever a powerfail recovery occurs. When an AST service entry point address is not specified, future power-recovery ASTs do not occur until an AST entry point is again specified.

Macro Call:

```
SPRA$   [ast]

  ast  =  AST service routine entry point address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ITS  --  AST entry point address is already unspecified
IE.AST  --  Directive was issued from an AST service routine
            or ASTs are disabled
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Set Power Recovery AST system service for the image issuing the SPECIFY POWER RECOVERY AST directive.

2. ASTs are disabled while the AST service routine executes. They remain disabled until the service routine issues an AST SERVICE EXIT directive.

3. The process enters the powerfail AST service routine with the task stack in the following state:

   ```
   SP+06,12 - 0
   SP+04    - PS of process prior to AST
   SP+02    - PC of process prior to AST
   SP+00    - DSW of process prior to AST
   ```

   No trap-dependent parameters accompany a power-recovery AST; therefore, the AST SERVICE EXIT directive can be executed with the stack in the same state as when the AST was effected.

4. This directive cannot be issued from an AST service routine or when ASTs are disabled.

SPWN$

4.2.40 SPWN$ - SPAWN

The SPAWN directive instructs the system to create a subprocess to execute a specified image or (optionally) to execute a command line.

If the command line is not specified, the specified image is invoked and executed by the subprocess; its TI device is the TI of the parent process.

If the command line is specified, a mailbox is created and the command line is written to it. Because TI is set equivalent to the mailbox, the command interpreter reads the command from the mailbox. Note that the command interpreter used is the one specified in the user authorization file (UAF).

When the created subprocess terminates, its exit status is delivered to the termination AST routine. A termination mailbox is created for this purpose.

Macro Call:

```
SPWN$   tname,,,[ugc],[umc],[efn],[east],[esb],[cmdlin],
        [cmdlen],[unum],[dnam]
```

tname	=	Name of image to be executed; also, name to be used for subprocess
ugc	=	Unused
umc	=	Unused
efn	=	Event flag to be set when subprocess terminates
east	=	Address of termination AST routine
esb	=	Address of exit status block, an 8-word buffer containing the exit status in the first word; the other words are unused
cmdlin	=	Address of command line to be executed; overrides image name specified in tname
cmdlen	=	Length of command line
unum	=	Unit number of TI device for process; if omitted, parent's TI is used
dnam	=	Device name of TI device for process; if omitted, parent's TI is used

DSW Return Codes:

IS.SUC	--	Succesful completion
IE.UPN	--	Insufficient dynamic memory (SS$_INSFMEM)
IE.INS	--	Task name invalid
IE.ACT	--	Process name already exists
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.ADP	--	Part of the DPB, exit status block, or command, line is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

4.2.41 SRDA$ - SPECIFY RECEIVE DATA AST

The SPECIFY RECEIVE DATA AST directive instructs the system to record either of the following conditions:

- That receive-data ASTs for the issuing image are required, and the address to which to transfer control when data has been placed in the image's mailbox (RCVDprocessname)
- That receive-data ASTs for the issuing task are no longer required

When the directive specifies an AST service routine entry point address, receive-data ASTs for the image occur whenever data has been placed in the image's mailbox (RCVDprocessname). VAX/VMS transfers control to the specified address.

When the directive omits an entry point address, VAX/VMS disables receive-data ASTs for the issuing image. Receive-data ASTs do not occur until the image issues another SPECIFY RECEIVE DATA AST directive that specifies an entry point address.

Macro Call:

```
SRDA$   [ast]

  ast  =  AST service routine entry point address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ITS  --  AST entry point address is already unspecified
IE.AST  --  Directive was issued from an AST service routine
            or ASTs are disabled
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. The task enters the receive-data AST service routine with the task stack in the following state:

   ```
   SP+06,12 - 0
   SP+04    - PS of process prior to AST
   SP+02    - PC of process prior to AST
   SP+00    - DSW of process prior to AST
   ```

   No trap-dependent parameters accompany a receive-data AST; therefore, the AST SERVICE EXIT directive must be executed with the stack in the same state as when the AST was effected.

2. This directive cannot be issued from an AST service routine or when ASTs are disabled.

3. VAX/VMS implements the SPECIFY RECEIVE DATA AST through the use of the set AST enable QIO I/O function for an unsolicited message to the mailbox. When a message is sent to the mailbox, an AST is given to the image. The AST is reenabled by a subsequent AST SERVICE EXIT directive.

4. Also refer to the section in this chapter on the RECEIVE DATA directive.

4.2.42 STLO$ - STOP FOR LOGICAL OR OF EVENT FLAGS

The STOP FOR LOGICAL OR OF EVENT FLAGS directive instructs the system to stop the execution of the issuing image until VAX/VMS sets one or more of the indicated local event flags from one of the following groups.

GR0 -- Flags 1 through 16
GR1 -- Flags 17 through 32

The process does not stop if any of the indicated flags is already set when it issues the directive.

This directive cannot be issued from an AST service routine.

A process that is stopped because none of the indicated event flags is set can be restarted only when one or more of the specified event flags is set. Such a stopped process cannot become unstopped by means of an UNSTOP TASK directive.

Macro Call:

```
STLO$   grp,msk

   grp  --  Event flag group
   msk  --  A 16-bit mask word
```

DSW Return Codes:

IS.SUC	--	Successful completion
IE.AST	--	Directive was issued from an AST service routine
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Wait for Logical OR of Event Flags system service on behalf of the image issuing the STOP FOR LOGICAL OR OF EVENT FLAGS directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63

STOP$S

4.2.43 STOP$S - STOP

The STOP directive instructs the system to place a process in a state of hibernation. A STOP directive can stop only the process that issues the directive, and this directive cannot be issued from an AST service routine.

A process stopped by the STOP directive can be restarted only by an UNSTOP TASK directive issued by one of its own ASTs or by another process.

Macro Call:

```
STOP$S
```

DSW Return Codes:

IE.SET	--	Successful completion
IE.INS	--	Process has no name
IE.AST	--	Directive was issued from an AST service routine
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Hibernate system service on behalf of the process issuing the STOP directive.

2. A stopped process retains control of the system resources allocated to it. VAX/VMS makes no attempt to free these resources.

3. VAX/VMS maintains an indicator to determine whether any wake requests have been issued for an active process. If the wake pending indicator is set and the process issues a hibernate request, the process remains active, and the wake pending indicator is cleared. A subsequent hibernate request causes the process to hibernate.

   Thus, if a STOP directive is issued by a process that has pending unstop requests, the following occurs:

   The status returned is success.

   The process remains active.

   The wake pending indicator is cleared.

4. A process stopped by use of the STOP directive can be restarted (by use of the UNSTOP TASK directive) only if its process name is specified; therefore, a process is not allowed to stop itself unless it has a process name.

4.2.44 STSE$ - STOP FOR SINGLE EVENT FLAG

The STOP FOR SINGLE EVENT FLAG directive instructs the system to stop the execution of the issuing image until the specified local event flag is set. If the flag is set when the directive is issued, image execution continues. This directive cannot be issued from an AST service routine.

A process that is stopped because a specified event flag is not set can be restarted only when that event flag is set. Such a stopped process cannot be restarted by means of an UNSTOP TASK directive.

Macro Call:

```
STSE$   efn

efn  =  Event flag number
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.AST  --  Directive was issued from an AST service routine
IE.IEF  --  Invalid event flag number because: (1) in the range
            33 to 64, but no associated common event flags; (2)
            in the range 65 to 96, but no associated group global
            event flags; (3) not in the range 1 to 96
IE.ADP  --  Part of the DPB is out of the issuing image's address
            space
IE.SDP  --  DIC or DPB size invalid
```

Notes:

1. VAX/VMS executes a Wait for Logical OR of Event Flags system service in behalf of the image issuing the STOP FOR SINGLE EVENT FLAG directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

SVDB$

4.2.45 SVDB$ - SPECIFY SST VECTOR TABLE FOR DEBUGGING AID

The SPECIFY SST VECTOR TABLE FOR DEBUGGING AID directive instructs the system to record the address of a table of SST service routine entry points for use by an intra-image debugging aid (ODT, for example).

To deassign the vector table, the parameters adr and len are omitted from the macro call.

When an SST service routine entry is specified in both the table used by the image and the table used by a debugging aid, the trap occurs for the debugging aid, not for the image.

Macro Call:

```
SVDB$   [adr][,len]

  adr  =  Address of SST vector table
  len  =  Length of (that is, number of entries in) the table in
          words
```

The vector table has the following format:

```
WD.  00  --  Odd address or nonexistent memory error
WD.  01  --  Memory protection violation
WD.  02  --  T-bit trap or execution of a BPT instruction
WD.  03  --  Execution of an IOT instruction
WD.  04  --  Execution of an illegal or reserved instruction
WD.  05  --  Execution of a non-RSX EMT instruction
WD.  06  --  Execution of a TRAP instruction
WD.  07  --  Not used
```

A table entry with a value of 0 indicates that the image will not process the corresponding SST.

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB or table is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

4.2.46 SVTK$ - SPECIFY SST VECTOR TABLE FOR TASK

The SPECIFY SST VECTOR TABLE FOR TASK directive instructs the system to record the address of a table of SST service routine entry points for use by the issuing image.

To deassign the vector table, the parameters adr and len are omitted from the macro call.

When an SST service routine entry is specified in both the table used by the image and the table used by a debugging aid, the trap occurs for the debugging aid, not for the image.

Macro Call:

```
SVTK$   [adr][,len]

  adr  =  Address of SST vector table
  len  =  Length of (that is, number of entries in) the table in
          words
```

The vector table has the following format:

```
WD.00  --  Odd address or nonexistent memory error
WD.01  --  Memory protection violation
WD.02  --  T-bit trap or execution of a BPT instruction
WD.03  --  Execution of an IOT instruction
WD.04  --  Execution of an illegal or reserved instruction
WD.05  --  Execution of a non-RSX EMT instruction
WD.06  --  Execution of a TRAP instruction
WD.07  --  Not used
```

A table entry with a value of 0 indicates that the image will not process the corresponding SST.

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB or table is out of the issuing
            image's address space
IE.SDP  --  DIC or DPB size is invalid
```

USTP$

4.2.47 USTP$ - UNSTOP TASK

The UNSTOP TASK directive instructs the system to awaken a process that is in a state of hibernation. An UNSTOP TASK directive restarts a specified process that has stopped itself by means of either a STOP directive or a RECEIVE DATA OR STOP directive. The UNSTOP TASK directive does not restart processes stopped for an event flag.

If the UNSTOP TASK directive is issued to a process that is executing an AST service routine and if that process was previously stopped by either a STOP directive or by a RECEIVE DATA OR STOP directive, the process becomes unstopped only when the execution of the AST service routine has been completed.

Macro Call:

```
USTP$   tname

tname  =  VAX/VMS process name
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.INS  --  Process name unknown
IE.PRI  --  Privilege violation (SS$_NOPRIV)
IE.UPN  --  Insufficient dynamic memory (SS$_INSFMEM)
IE.NOD  --  Image's quota exceeded (SS$_EXQUOTA)
IE.ADP  --  Part of the DPB is out of the issuing image's address
            space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Wake system service on behalf of the process issuing the UNSTOP TASK directive.

2. The process issuing an UNSTOP TASK directive must meet one of the following requirements:

   It must have the same UIC as the process to be unstopped.

   It must be in the same group as the process to be unstopped, and it must have the GROUP privilege.

   Otherwise, a process needs no privileges to issue an UNSTOP TASK directive. It is the responsibility of the unstopped process to determine whether it has been validly awakened.

3. VAX/VMS maintains an indicator to determine whether any wake requests have been issued for an active process. If the indicator is set and the process issues a hibernate request, the process remains active and the indicator is cleared. A subsequent hibernate request causes the process to hibernate.

Thus, if a STOP directive is issued by a process that has pending unstop requests, the following occurs:

The status returned is success.

The process remains active.

The wake-pending indicator is cleared.

4. If an UNSTOP TASK directive is issued for an image that is active, the status returned is a success. The process remains active.

WSIG$S

4.2.48 WSIG$S - WAIT FOR SIGNIFICANT EVENT

Because significant events do not exist in VAX/VMS, the WAIT FOR SIGNIFICANT EVENT directive does not correspond to any VAX/VMS system service request. No wait occurs when the directive is issued.

Macro Call:

```
WSIG$S  [err]

  err  =  Error routine address
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

4.2.49 WTLO$ - WAIT FOR LOGICAL OR OF EVENT FLAGS

The WAIT FOR LOGICAL OR OF EVENT FLAGS directive instructs the system to block the execution of the issuing image until VAX/VMS sets an indicated event flag from one of the following groups.

```
GR 0  --  Flags   1 to 16
GR 1  --  Flags  17 to 32
GR 2  --  Flags  33 to 48
GR 3  --  Flags  49 to 64
GR 4  --  Flags  65 to 80
GR 5  --  Flags  81 to 96
```

The process does not wait if any of the indicated flags is already set when it issues the directive.

Macro Call:

```
WTLO$   grp,msk

  grp  =  Event flag group
  msk  =  A 16-bit flag mask word
```

DSW Return Codes:

IS.SUC	--	Successful completion
IE.IEF	--	Invalid event flag number because: (1) in the range 33 to 64, but no associated common event flags; (2) in the range 65 to 96, but no associated group global event flags; (3) not in the range 1 to 96
IE.ADP	--	Part of the DPB is out of the issuing image's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. VAX/VMS executes a Wait for Logical OR of Event Flags system service on behalf of the image issuing the WAIT FOR LOGICAL OR OF EVENT FLAGS directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

4. The DSW status IE.IEF is returned if an image that does not have a common event flag cluster associated with it attempts to wait for flags 33 through 64.

WTSE$

4.2.50 WTSE$ - WAIT FOR SINGLE EVENT FLAG

The WAIT FOR SINGLE EVENT FLAG directive instructs the system to block the execution of the issuing image until the indicated event flag is set. If the flag is set when the directive is issued, image execution continues.

Macro Call:

```
WTSE$   efn

  efn  =  Event flag number
```

DSW Return Codes:

```
IS.SUC  --  Successful completion
IE.IEF  --  Invalid event flag number because:  (1) in the
            range 33 to 64, but no associated common event
            flags; (2) in the range 65 to 96, but no
            associated group global event flags; (3) not in
            the range 1 to 96
IE.ADP  --  Part of the DPB is out of the issuing image's
            address space
IE.SDP  --  DIC or DPB size is invalid
```

Notes:

1. VAX/VMS executes a Wait for Logical OR of Event Flags system service in behalf of the image issuing the WAIT FOR SINGLE EVENT FLAG directive.

2. A task image must be associated with common or group global event flags to access flags in the common or group global clusters (see Section 2.5).

3. Event flag conversion is as follows:

Cluster	RSX-11M	VAX/VMS
Local	1 to 32	32 to 63
Common	33 to 64	64 to 95
Group global	65 to 96	96 to 127

CHAPTER 5

I/O DRIVERS

VAX/VMS images request services directly from I/O drivers and ACPs by issuing Queue I/O Request macroinstructions. Each macroinstruction consists of the following types of arguments:

- An I/O function code
- Function-independent parameters, for example, I/O channel and event flag number
- Function-dependent parameters P1 through P6

VAX/VMS I/O function code names have the following format:

```
IO$_function
```

Many function codes have subfunction modifiers that can be associated with them. Subfunction modifier names have the following format:

```
IO$M_subfunction
```

The following are examples of VAX/VMS function codes and subfunction modifiers:

```
IO$_WRITELBLK
IO$_READPROMPT!IO$M_NOFILTR
IO$_READVBLK
IO$_DELETE!IO$M_DELETE
```

When an RSX-11M image running under VAX/VMS issues a QUEUE I/O REQUEST directive, VAX/VMS determines the equivalent native function and executes a Queue I/O Request system service on behalf of the image. The I/O request is processed by the VAX/VMS I/O system and the function is performed by a standard VAX/VMS device driver or ACP. Usually, RSX-11M I/O requests correspond to similar VAX/VMS requests. As a result, the RSX-11M image is not aware of any differences in the I/O systems. However, if an image issues an I/O request that depends on characteristics of the RSX-11M I/O system that are not present in the VAX/VMS I/O system, the requested I/O operation may not occur exactly as expected. In that event, the user should consult this chapter.

Each RSX-11M I/O request consists of a function code, function-independent parameters, and function-dependent parameters. When VAX/VMS receives a QUEUE I/O REQUEST directive, it forms the equivalent VAX/VMS arguments for each RSX-11M parameter specified in the directive. Because VAX/VMS issues queue I/O requests using the VAX/VMS I/O system, it must convert RSX-11M queue I/O requests to the native format for processing by the appropriate driver or ACP.

VAX/VMS handling of RSX-11M function-independent parameters, for example, efn, lun, and ast, is described in Chapters 2 and 3 and in

the description of the QUEUE I/O REQUEST directive in Chapter 4. This chapter describes how VAX/VMS handles I/O function codes and I/O function-dependent parameters.

5.1 SUPPORTED DEVICES

VAX/VMS supports RSX-11M I/O functions for devices supported by both RSX-11M and VAX/VMS; that is, for disks, terminals, line printers, card readers, magnetic tapes, and the null device. The VAX/VMS I/O User's Guide lists the devices supported by VAX/VMS.

If an RSX-11M image performs I/O to a device that VAX/VMS does not support and that does not require special-case software, the I/O request is handled as if it specified a disk device. The I/O function code and parameters (P1 through P6) are handled just as they are for disk. No subfunction bits are used.

5.2 GET LUN INFORMATION DIRECTIVE

The GET LUN INFORMATION directive returns the same device-independent information under VAX/VMS as it does under RSX-11M Version 3.2. The format of the information returned for all devices is presented in the description of the GET LUN INFORMATION directive in Chapter 4. The VAX/VMS I/O User's Guide describes the format of the device-dependent information returned.

5.3 STANDARD I/O FUNCTIONS

The standard RSX-11M I/O functions -- attach, detach, and cancel I/O; read and write virtual block; and read and write logical block -- are supported for all devices in VAX/VMS. The sections that follow provide additional information about attach, detach, and cancel I/O.

5.3.1 Attach and Detach I/O Device (IO.ATT and IO.DET)

VAX/VMS categorizes devices as shareable and nonshareable. A shareable device, such as a disk, can be accessed by many users without affecting the integrity of the data. A nonshareable device, such as a terminal, allows access from only one process at a time. When an image assigns a channel to a nonshareable device, VAX/VMS implicitly allocates the device for exclusive use by the process. In the RSX-11M sense, the system attaches the device for the process. Because VAX/VMS performs implicit allocation, images do not have to explicitly allocate and deallocate nonshareable devices during execution.

If an image must have exclusive access to a shareable device, the device can be allocated in either of two ways:

1. By an image issuing an Allocate Device system service
2. By a user typing an ALLOCATE command to the MCR or DCL command interpreter

Using the ALLOCATE command has an advantage over the Allocate Device system service. It eliminates the need for error recovery by the image if the device is not available for allocation.

Tasks running in RSX-11M frequently attach terminals and other devices to prevent another task from using them. These devices are shareable in an RSX-11M system. When an RSX-11M image running under VAX/VMS issues a QUEUE I/O REQUEST to attach a device, VAX/VMS performs no operation and returns a success status to the image. If the target device is nonshareable, VAX/VMS allocates the device when the image assigns a LUN to it. In effect, therefore, the device is attached. If the device is shareable, it remains unallocated after the directive status is returned. When an RSX-11M image requires allocation of a shareable device, the device must be allocated from a terminal or an indirect command file by using an ALLOCATE command.

The RSX-11M function code IO.ATT and IO.DET have meaning for terminals under VAX/VMS, as described in Section 5.8, "Terminal Driver." For example, issuing an attach or detach request causes a cancel CTRL/O function.

5.3.2 Cancel I/O Requests (IO.KIL)

When an RSX-11M image issues a kill I/O request for a VAX/VMS device, VAX/VMS executes a Cancel I/O on Channel system service. This system service cancels all I/O issued from the designated channel. This differs from the RSX-11M approach in that RSX-11M causes all I/O from the issuing task to the device to be canceled. When a cancel I/O request is issued for a disk device, no operation is performed. VAX/VMS returns a success status to the image.

When the Cancel I/O on Channel system service executes, it notifies the driver immediately. Queued I/O requests are canceled immediately; however, I/O that the driver is currently processing is not necessarily canceled.

5.4 I/O STATUS BLOCK AND STATUS RETURNS

When VAX/VMS completes an I/O operation, it returns a code indicating the status of the request in an I/O status block. When an RSX-11M image issues a request, VAX/VMS returns status information in an I/O status block that has the standard RSX-11M format, as illustrated in Figure 5-1.

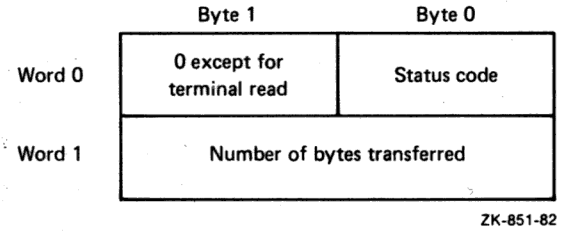


Figure 5-1: Format of RSX-11M I/O Status Block under VAX/VMS

The return code can be IS.SUC or any of the error status codes listed in Table 5-1. The status code IS.SUC corresponds to the VAX/VMS status code SS$_NORMAL. VAX/VMS equivalents for RSX-11M error codes also are provided in Table 5-1.

The high-order byte of word 0 always contains a 0 except in the case of terminal I/O read requests. For a terminal read request, that byte indicates the line terminator, as described in Section 5.8.14.

The second word of the I/O status block contains the number of bytes read or written.

Table 5-1: I/O Status Return Codes

DSW Code	VAX/VMS Code	Meaning
IE.ABO	SS$_ABORT	An I/O request was canceled before the operation was completed, or a network link was broken.
	SS$_CANCEL	An I/O request was canceled before the operation was completed.
IE.ALN	SS$_FILALRACC	A DECnet-VAX logical link already existed or was pending.
IE.BAD	SS$_BADPARAM	A call to a network or file ACP contained invalid parameters.
IE.BCC	SS$_DATACHECK	A data check found a mismatch between disk data and memory data.
IE.BDR	SS$_BADIRECTORY	A file specified as a directory either was not a directory or contained invalid data.
IE.BDV	SS$_NOTFILEDEV	A file specification contained references to a directory or a file on a device that was not file-structured.
IE.BHD	SS$_BADFILEHDR	A file header contained invalid data; for example, the structure was not consistent or the storage map indicated free blocks.
	SS$_FILESTRUCT	The file structure on an accessed volume was invalid for the called ACP.
IE.BLK	SS$_ILLBLKNUM	The logical block number specified for a file did not exist on disk.
IE.BNM	SS$_BADFILENAME	A file name contained illegal characters or was longer than nine characters.
IE.BVR	SS$_BADFILEVER	A file version number was greater than 32767.
	SS$_TOOMANYVER	The maximum number of versions for a file already existed and all had greater version numbers than the specified version number.
IE.CKS	SS$_BADCHKSUM	The checksum in a file header was invalid.
IE.CLO	SS$_FILELOCKED	A process attempted to access a locked file.

(continued on next page)

Table 5-1 (Cont.): I/O Status Return Codes

DSW Code	VAX/VMS Code	Meaning
IE.CNR	SS$_REJECT	A request to connect to an object at a remote network node failed.
IE.DAA	SS$_DEVALLOC	An allocate request specified a device already allocated to another user.
IE.DAO	SS$_BUFFEROVF	A buffer was not large enough for a string output by a system service; the string was truncated.
	SS$_DATAOVERUN	A buffer was not large enough for data output by a system service.
	SS$_MBTOOSML	A mailbox was too small for data sent to it.
IE.DFU	SS$_DEVICEFULL	A device was full or did not have enough contiguous blocks to fill a request.
	SS$_DIRFULL	A file could not be created because the specified directory was full.
IE.DNA	SS$_DEVNOTALLOC	A deallocate request specified a device that was not allocated.
	SS$_VOLINV	The volume-valid bit for a requested volume was not set.
IE.DNR	SS$_DEVNOTMOUNT	A dismount request specified a device that was not mounted.
	SS$_VOLINV	A volume-valid bit for a requested volume was not set.
IE.DSQ	SS$_EXDISKQUOTA	A process exceeded its disk quota.
IE.DUP	SS$_DUPFILENAME	A specified file already existed in the specified directory.
IE.EOF	SS$_ENDOFFILE	The end-of-a-file was reached. This was an EOF mark on a tape, an EOF card in a card reader, an empty mailbox, or the end of virtual memory.
IE.EOT	SS$_BEGOFFILE	A backspace operation reached the beginning of a file.
	SS$_ENDOFTAPE	The end-of-tape mark was encountered on a tape.
IE.EOV	SS$_ENDOFVOLUME	The end-of-a-volume was encountered.
IE.EXP	SS$_FILNOTEXP	A file could not be written or deleted because it had not reached its expiration date.

(continued on next page)

Table 5-1 (Cont.): I/O Status Return Codes

DSW Code	VAX/VMS Code	Meaning
IE.FHE	SS$_CRTLERR	A hardware controller failed during an I/O operation.
	SS$_DRVERR	A device driver failed during an I/O operation.
	SS$_UNSAFE	A device driver was unusable.
IE.HFU	SS$_HEADERFULL	A file-header map was full and its extension was inhibited.
IE.IES	SS$_BADESCAPE	A terminal escape sequence was invalid.
IE.IFC	SS$_ILLCNTRFUNC	The control function specified for an ACP was invalid.
	SS$_ILLIOFUNC	The function code specified for an explicit I/O request was invalid.
IE.LCK	SS$_ACCONFLICT	The access protection for a file did not allow a requested access.
IE.NDR	SS$_NOLINKS	A logical network link could not be created because no more slots were available.
IE.NLN	SS$_FILNOTACC	No file had been accessed on a channel that was specified for an I/O operation.
IE.NNN	SS$_NOSUCHNODE	A specified network node did not exist.
IE.NOD	SS$_ACPVAFUL	A file ACP could not access a volume because it had no more virtual memory for volume service.
	SS$_EXQUOTA	A process attempted to exceed its limit or quota for a resource.
	SS$_INSFMEM	More system dynamic memory was required than was available.
	SS$_INSFWSL	A process required more pages in its working set than it was allowed.
IE.NSF	SS$_NOMOREFILES	No more files matching a wild card specification existed; at least one matching file was previously found.
	SS$_NOSUCHFILE	A specified file did not exist.
IE.OFL	SS$_DEVOFFLINE	A device went off-line.
	SS$_MEDOFL	A requested device had no medium mounted on it (such as a tape or a disk).

(continued on next page)

Table 5-1 (Cont.): I/O Status Return Codes

DSW Code	VAX/VMS Code	Meaning
IE.PES	SS$_PARTESCAPE	A terminal escape sequence was truncated at the end of its buffer; the remainder of the sequence was written in the type-ahead buffer.
IE.PRI	SS$_NOPRIV	A process requested initialization of a volume that it did not have the privilege to write.
IE.RER	SS$_FCPREADERR	An error occurred in reading file control data (such as a directory).
	SS$_FCPREWNDERR	An error occurred in rewinding a volume.
	SS$_FCPSPACERR	An I/O error occurred while a file control primitive was skipping spaces within or among files.
IE.RSU	SS$_MBFULL	A mailbox could not accept another message because it was full.
	SS$_VECINUSE	The CTRL/C vector for a process was already in use by a controlling process.
IE.SNC	SS$_FILENUMCHK	The index file for a volume contained invalid data.
IE.SPC	SS$_ACCVIO	A process attempted to access memory outside its virtual address space. The PC contains the address of the invalid instruction.
IE.SQC	SS$_FILESEQCHK	The file sequence number in a file header was invalid; the directory entry pointed to an obsolete or deleted file.
IE.TMO	SS$_TIMEOUT	An input operation was not completed within the specified timeout period.
IE.VER	SS$_PARITY	A device-dependent error occurred.
IE.WAC	SS$_ACCONFLICT	The access protection for a file did not allow a requested access.
IE.WAT	SS$_BADATTRIB	An invalid attribute was specified for a read or write using a file ACP.
IE.WER	SS$_FCPWRITERR	An I/O error occurred while a file control primitive was writing.
IE.WLK	SS$_WRITLCK	The hardware write-lock switch was set on a requested disk.

(continued on next page)

Table 5-1 (Cont.): I/O Status Return Codes

DSW Code	VAX/VMS Code	Meaning
IS.PND	None	An I/O request is pending.
IS.RDD	SS$_RDDELDATA	Successful read of a physical block that contains deleted data marks (see Table 5-2, Write Physical Block with Deleted Data Mark).
IS.SUC	SS$_NORMAL	An operation succeeded.

5.5 DISK DRIVER

Table 5-2 provides the correspondence between RSX-11M disk function codes and VAX/VMS disk function codes.

Table 5-2: Disk Function Code Correspondence

Function	RSX-11M Code	VAX/VMS Code or Action
Attach Device	IO.ATT	No operation
Detach Device	IO.DET	No operation
Cancel I/O Requests	IO.KIL	No operation
Read Logical Block	IO.RLB	IO$_READLBLK
Write Logical Block	IO.WLB	IO$_WRITELBLK
Read Virtual Block	IO.RVB	IO$_READVBLK
Write Virtual Block	IO.WVB	IO$_WRITEVBLK
Read Physical Block	IO.RPB	IO$_READPBLK
Write Physical Block	IO.WPB	IO$_WRITEPBLK
Write Physical Block with Deleted Data Mark	IO.WDD	IO$_WRITEPBLK!IO$M_DELDATA (Only for RX01 and RX02 diskettes.)
Load Overlay	IO.LOV	Special form of IO.RLB performed only on OV (overlay device); An IO.RVB is performed on LUNs not assigned to OV.
Pack Acknowledge	IO.STC	IO$_PACKACK

Table 5-3 provides the correspondence of RSX-11M function-dependent parameters to VAX/VMS arguments.

Table 5-3: Disk Parameter Correspondence

Parameter Function	RSX-11M Pn	VAX/VMS Pn
Starting buffer address	P1	P1 (stadd)
Buffer size (size)	P2	P2
High block number (bklh)	P4	P3 (high half of longword)
Low block number (blkl)	P5	P3 (low half of longword)

5.6 MAGNETIC TAPE DRIVER

Table 5-4 provides the correspondence between RSX-11M magnetic tape function codes and VAX/VMS magnetic tape function codes.

Table 5-4: Magnetic Tape Function Code Correspondence

Function	RSX-11M Code	VAX/VMS Code or Action
Attach Device	IO.ATT	No operation
Detach Device	IO.DET	No operation
Cancel I/O Requests	IO.KIL	Cancel I/O on Channel system service
Read Logical Block	IO.RLB	IO$_READLBLK
Write Logical Block	IO.WLB	IO$_WRITELBLK
Read Virtual Block	IO.RVB	IO$_READVBLK
Write Virtual Block	IO.WVB	IO$_WRITEVBLK
Write End-of-File Mark	IO.EOF	IO$_WRITEOF
Read Logical Block Reverse	IO.RLV	IO$_READPBLK!IO$M_REVERSE
Rewind Unit	IO.RWD	IO$_REWIND
Rewind and Turn Unit Off Line	IO.RWU	IO$_REWINDOFF
Mount Tape and Set Characteristics	IO.SMO	IO$_SETMODE (only parity and density can be set)
Sense Tape Characteristics	IO.SEC	IO$_SENSEMODE
Space Blocks	IO.SPB	IO$_SPACERECORD
Space Files	IO.SPF	IO$_SPACEFILE
Set Tape Characteristics	IO.STC	IO$_SETMODE (only parity and density can be set)

Table 5-5 provides the correspondence of RSX-11M function-dependent parameters to VAX/VMS arguments.

Table 5-5: Magnetic Tape Parameter Correspondence

Parameter Function	RSX-11M Pn	VAX/VMS Pn
Starting buffer address (stadd)	P1	P1
Buffer size (size)	P2	P2
Characteristic bits (cb) of IO.SMO and IO.STC	P1	P1
Number of blocks to space past (nbs) of IO.SPB	P1	P1
Number of EOFs to space past (nes) of IO.SPF	P1	P1

5.7 LINE PRINTER DRIVER

Table 5-6 provides the correspondence between RSX-11M line printer function codes and VAX/VMS function codes or resultant action.

Table 5-6: Line Printer Function Code Correspondence

Function	RSX-11M Code	VAX/VMS Code or Action
Attach Device	IO.ATT	No operation
Detach Device	IO.DET	No operation
Cancel I/O Requests	IO.KIL	Cancel I/O on Channel system service
Write Logical Block	IO.WLB	IO$_WRITELBLK
Write Virtual Block	IO.WVB	IO$_WRITEVBLK
Write Physical Block	IO.WPB	IO$_WRITEPBLK

Table 5-7 provides the correspondence of RSX-11M function-dependent parameters to VAX/VMS arguments.

Table 5-7: Line Printer Parameter Correspondence

Parameter Function	RSX-11M Pn	VAX/VMS Pn
Starting buffer address (stadd)	P1	P1
Buffer size (size)	P2	P2
Vertical format control character (vfc)	P3	P4

When using VAX/VMS line printers, keep in mind the following points:

- VAX/VMS line printers are not shareable. VAX/VMS implicitly allocates a line printer when a channel is assigned.
- VAX/VMS line printers normally are spooled. A spooled printer is allocated to the print symbiont. VAX/VMS does not allow a process to allocate a spooled device unless it has the privilege to do so. An RSX-11M image is not allowed exclusive use of a spooled device (for example, printer) unless the process in which it is running has the necessary privilege and the ALLOCATE command (MCR or DCL) has been issued to reserve the device prior to image execution.
- If a printer is allocated or not spooled, the RSX-11M image's IO.WLB and IO.WVB requests for it produce exactly the same results as in the RSX-11M operating system.
- See Section 3.10 for a discussion of the requirements for issuing IO.WLB and IO.WVB requests to a spooled device.
- If an RSX-11M image issues a GET LUN INFORMATION directive for a spooled device, the information returned is that for the intermediate device.

5.8 TERMINAL DRIVER

Table 5-8 provides the correspondence of RSX-11M function codes to VAX/VMS functions. Table 5-9 provides the correspondence of the RSX-11M function-dependent parameters P1 through P6 to their VAX/VMS equivalents for terminal devices. Table 5-10 lists the subfunction bits applicable for each RSX-11M function code and provides notes describing VAX/VMS handling of these subfunctions for terminals.

VAX/VMS places restrictions on the I/O functions that can be performed on TI, CO, and CL because they are mapped to process-permanent files. It places the same restrictions on I/O to user-created process-permanent files. Section 5.8.15 describes these restrictions.

Table 5-8: Terminal Function Code Correspondence

Function	RSX-11M Code	VAX/VMS Code or Action
Attach Device	IO.ATT	Terminal not attached; forces cancel CTRL/O on next write
Detach Device	IO.DET	Terminal not detached; forces cancel CTRL/O on next write
Cancel I/O Requests	IO.KIL	Cancel I/O on Channel system service
Read Logical Block	IO.RLB	IO$_READLBLK
Write Logical Block	IO.WLB	IO$_WRITELBLK
Read Virtual Block	IO.RVB	IO$_READVBLK
Write Virtual Block	IO.WVB	IO$_WRITEVBLK
Read Physical Block	IO.RPB	IO$_READPBLK
Write Pass All	IO.WAL	IO$_WRITEPBLK
Read Logical Block after Prompt	IO.RPR	IO$_READPROMPT
Get Multiple Characteristics	SF.GMC	IO$SENSEMODE
Set Multiple Characteristics	SF.SMC	IO$_SETMODE
Get Terminal Support	IO.GTS	Standard data returned

Table 5-9: Terminal Parameter Correspondence

Parameter Function	RSX-11M Pn	VAX/VMS Pn
Starting buffer address (stadd)	P1	P1
Buffer size (size)	P2	P2
Vertical format control character (vfc) on write	P3	P4[1]
Timeout count (tmo) on read with prompt	P3	P3
Prompt address (pradd) for read with prompt	P4	P5
Prompt size (prsize) for read with prompt	P5	P6
Vertical format control character (vfc) for read with prompt	P6	none

1. For all read functions except IO.RPB and IO.RST, the VAX/VMS P4 parameter specifies RETURN, ESCAPE, and CTRL/Z as terminators. For IO.RPB, no characters are terminators. For IO.RST, P4 is 0 specifying that all characters with a value less than an ASCII space are terminators except form feed, vertical tab, backspace, delete, and horizontal tab.

NOTE

The remaining device-specific function codes are the equivalent of the logical OR of a subfunction bit and one of the standard function codes IO.ATT, IO.RLB, or IO.WLB. See Table 5-10.

Table 5-10: Subfunction Bit Correspondence

FUNCTION	EQUIVALENT WITH SUBFUNCTION BIT	APPLICABLE SUBFUNCTION BITS X = Corresponds directly to VAX/VMS function. n = Indicates correspondingly-numbered note. TF.AST	TF.BIN	TF.CCO	TF.ESQ	TF.RAL	TF.RNE	TF.RST	TF.WAL	TF.WBT	TF.XOF
STANDARD FUNCTIONS:											
IO.ATT		1			2						
IO.DET											
IO.KIL											
IO.RLB						3	X	X			
IO.RVB						4	4	4			
IO.RPB							X				
IO.WLB				X					X	2	
IO.WVB				X					4	4	
IO.WPB											
DEVICE-SPECIFIC FUNCTIONS:											
IO.ATA	IO.ATT!TF.AST (see Note 1)				2						
IO.CCO	IO.WLB!TF.CCO								X	2	
SF.GMC											
IO.GTS											
IO.RAL	IO.RLB!TF.RAL (see Note 3)						X	X			
IO.RNE	IO.RLB!TF.RNE					3		X			
IO.RPR			2			3	X	X			2
IO.RST	IO.RLB!TF.RST					3	X				
SF.SMC											
IO.WAL	IO.WLB!TF.WAL			X						2	
IO.WBT	IO.WLB!TF.WBT (See Note 2)			X					X		

NOTES: 1. No attach performed. Enables for CTRL/C ASTs only. See Section 5.8.1.

2. Subfunction bit ignored for one of the following reasons.

Function	Reason
TF.ESQ, TF.XON	These are characteristics of the terminal line and cannot be controlled on a per-request basis.
TF.WBT	The write breakthrough function is not supported in VAX/VMS. See Section 5.8.8.
TF.BIN	Function is not supported in VAX/VMS.

3. Sets the VAX/VMS function modifier IO$M_NOFILTR. See Section 5.8.4.1.

4. RSX-11M virtual functions do not accept these subfunction bits.

Two differences between the VAX/VMS compatibility mode terminal driver and the RSX-11M terminal driver can affect applications that display or request input from a terminal. These differences concern the <CR> and <LF> control characters sent to the terminal, as follows:

1. When a compatibility mode image displays a message using a terminal write request, a difference occurs only if both of the following conditions occur:

   - The last character written or echoed to the terminal was an <LF> character.
   - The first character of the next write request is an <LF> character.

   If both conditions occur, VAX/VMS skips (does not send) the leading <LF> character in the next write request. The fact that VAX/VMS sends one, not two, <LF> characters changes the appearance of the output as seen by the terminal user. RSX-11M sends both <LF> characters under the same conditions.

2. When a compatibility mode image requests input using a terminal read request, a difference always occurs when the terminal user ends the read request by pressing the RETURN key, which generates a <CR> character. VAX/VMS automatically echoes the <CR> character plus a <LF> character to the terminal screen, whereas RSX-11M only echoes a <CR> character. Again, the terminal user would notice the difference on the output.

However, when the image executes a terminal read request terminated by a <CR> character that is followed by a terminal write request with a leading <LF> character, the two differences cancel each other, thus giving the screen the same appearance under VAX/VMS as under RSX-11M.

5.8.1 IO.ATT Function

When an RSX-11M image uses the IO.ATT function for a terminal, VAX/VMS performs no operation to alter the attached/detached status of the terminal, as described in Section 5.3.1. VAX/VMS does, however, issue a request to the terminal driver to cancel CTRL/O on the next operation to the terminal if that operation is a write. The VAX/VMS terminal driver subfunction modifier to cancel CTRL/O is IO$M_CANCTRLO. The RSX-11M terminal driver also forces a cancel CTRL/O (TF.CCO) on a write operation that follows an attach operation.

An IO.ATT function issued for TI, CO, or CL becomes a no-op.

5.8.1.1 IO.ATT!TF.AST and IO.ATA Functions - In VAX/VMS, an image can enable itself to receive an AST unsolicited character and a CTRL/C from the terminal. When the AST occurs, the image can respond to the unsolicited character or CTRL/C.

The RSX-11M function codes IO.ATT!TF.AST and IO.ATA are equivalent. When an RSX-11M image executing in VAX/VMS issues either of these codes, VAX/VMS issues a request to the terminal driver to enable the image for an unsolicited character or CTRL/C AST, depending on the values of parameters P1 and P3, respectively.

5.8.1.2 IO.ATT!TF.ESQ Function - In VAX/VMS, certain features that are characteristic of a terminal line are set by issuing a set terminal mode request (IO$_SETMODE) to the driver. Terminal characteristics cannot be altered for the duration of an I/O request by specifying a modifier to the request; nor can they be modified as a function of terminal allocation. The ability to recognize escape sequences on a terminal line is a characteristic of the terminal and must be set using IO$_SETMODE or a SET TERMINAL command using MCR or DCL.

In RSX-11M, the subfunction bit TF.ESQ is used with either of the attach function codes (IO.ATT or IO.ATA) to indicate that the image recognizes any escape sequences generated at the designated terminal. When VAX/VMS receives an I/O request containing the TF.ESQ subfunction from an RSX-11M image, it ignores that subfunction bit. The terminal characteristics remain unaltered.

To enable for escape sequences, an RSX-11M image should issue a set multiple characteristics request (SF.SMC).

5.8.2 IO.DET Function

When an RSX-11M image uses the IO.DET function for a terminal, VAX/VMS performs no operation to alter the attached/detached status of the terminal, as described in Section 5.3.1. VAX/VMS does, however, issue a request to the terminal driver to cancel CTRL/O on the next operation to the terminal if that operation is a write. The VAX/VMS terminal driver subfunction modifier to cancel CTRL/O is IO$M_CANCTRLO. The RSX-11M terminal driver also forces a cancel CTRL/O (TF.CCO) on a write operation that follows a detach operation.

An IO.DET function issued for TI, CO, or CL becomes a no-op.

5.8.3 IO.KIL function

An IO.KIL function issued for TI, CO, or CL becomes a no-op. That is, it is ignored.

5.8.4 IO.RLB, IO.RAL, IO.RNE, IO.RST, and IO.RTT Functions

The function codes IO.RLB, IO.RAL, IO.RNE, and IO.RST all allow an image to read a logical block from a terminal. When VAX/VMS receives a read-logical-block request from an RSX-11M image, it issues an IO$_READLBLK request on behalf of the image. There is a direct correspondence between IO.RLB and IO$_READLBLK. The RSX-11M function codes IO.RAL, IO.RNE, and IO.RST are the equivalents of the logical OR of IO.RLB and a subfunction bit. The following sections describe VAX/VMS handling of subfunction bits used with read-logical-block requests.

5.8.4.1 IO.RLB!TF.RAL and IO.RAL - In VAX/VMS, the default terminal driver operation on a read request is to intercept and interpret control characters, for example, TAB, CTRL/R, CTRL/U, and DELETE. However, a native image has two options for restricting the interception of control characters by the driver.

- It can specify the subfunction modifier IO$M_NOFILTR on a read function (either IO$_READLBLK, IO$_READVBLK, or IO$_READPROMPT) to prevent the driver from intercepting CTRL/U, CTRL/R, or DELETE.

- It can issue a read-physical-block (IO$_READPBLK) request to prevent the driver from interpreting any characters.

Normally, an RSX-11M image that issues a read-passing-all-data request actually wants to receive only a subset of the possible control characters; that is, it wants to receive CTRL/U, CTRL/R, and DELETE. As a result, when VAX/VMS receives an IO.RLB!TF.RAL or IO.RAL request from an RSX-11M image, it issues a request specifying IO$_READLBLK!IO$M_NOFILTR on behalf of the image.

The VAX/VMS equivalent of the RSX-11M read-passing-all-data function is read physical block (IO$_READPBLK). IO$_READPBLK corresponds directly to the RSX-11M function code IO.RPB. IO.RPB has been added to the function codes that can be issued by an RSX-11M image to allow execution of the read-passing-all-data function under VAX/VMS. An RSX-11M image that a read-passing-all-data function under VAX/VMS must be modified to issue an IO.RPB. An image issuing IO.RPB under the RSX-11M operating system runs without receiving an error; that is, IO.RPB is a legal function.

NOTE

In RSX-11M, an IO.RPB request is equivalent to an IO.RLB request with a subfunction bit set. IO.RAL or IO.RPB work on both VAX/VMS and RSX-11M systems.

VAX/VMS requires the image to have the appropriate privilege to read a physical block.

5.8.4.2 IO.RLB!TF.RNE and IO.RNE Functions - The RSX-11M function codes IO.RLB!TF.RNE and IO.RNE are equivalent. Either one corresponds directly to the VAX/VMS function code IO$_READLBLK or IO$_READVBLK with a no echo function modifier (IO$M_NOECHO).

5.8.4.3 IO.RLB!TF.RST and IO.RST Functions - The RSX-11M function codes IO.RLB!TF.RST and IO.RST are equivalent. Either one corresponds directly to the VAX/VMS function code IO$_READLBLK with a function modifier of IO$M_TRMNOECHO and a record-termination parameter (P4) of 0. IO$M_TRMNOECHO prevents echoing of the line terminator. A record termination parameter of 0 causes all characters with a value less than an ASCII space to be terminators except form feed, vertical tab, backspace, and horizontal tab.

5.8.4.4 IO.RLB!TF.RTT and IO.RTT Functions - The RSX-11M function codes IO.RLB!TF.RTT and IO.RTT are equivalent. Either one corresponds directly to the VAX/VMS function code IO$_READVBLK with a function modifier of IO$M_TRMNOECHO and a record-termination parameter P4.

5.8.5 IO.RPR Function

The IO.RPR function code corresponds directly to the VAX/VMS IO$_READPROMPT function code. However, the RSX-11M P6 parameter (vertical control character) is ignored. VAX/VMS handling of the subfunction bits TF.RAL, TF.RNE, and TF.RST with IO.RPR is exactly the same as it is for IO.RLB. VAX/VMS does not support use of the subfunction bit TF.BIN. VAX/VMS also ignores the subfunction bit TF.XOF if it is specified.

If an IO.RPR function is issued for TI, CO, or CL and these devices correspond to process permanent files, the prompt is ignored.

5.8.5.1 IO.RPR!TF.XOF Function - Under VAX/VMS, certain features that are characteristic of a terminal line are set by issuing a set-terminal-mode request (IO$_SETMODE) to the driver. A subfunction modifier indicates the characteristic to be changed. Terminal characteristics cannot be altered for the duration of an I/O request by specifying a modifier to the request; nor can they be modified as a function of terminal allocation. The ability to control XON/XOFF on a terminal line is a characteristic of the terminal and must be set using IO$_SETMODE.

In RSX-11M, the subfunction bit TF.XOF is used with IO.RPR to control XON/XOFF at the designated terminal. When VAX/VMS receives an I/O request containing the TF.XOF subfunction from an RSX-11M image, it ignores that subfunction bit. The terminal characteristics remain unaltered.

To control XON/XOFF, an RSX-11M image should issue a set multiple characteristics request (SF.SMC).

5.8.6 IO.RVB Function

The IO.RVB function code corresponds directly to the VAX/VMS IO$_READVBLK. No subfunction bits are supported in RSX-11M for IO.RVB.

5.8.7 IO.RPB Function

See the discussion of IO.RAL in Section 5.8.4.1.

5.8.8 IO.WLB, IO.CCO, and IO.WBT Functions

The function codes IO.WLB, IO.CCO, and IO.WBT all allow an image to write a logical block to a terminal. When VAX/VMS receives a write-logical-block request from an RSX-11M image, it issues an IO$_WRITELBLK request. There is a direct correspondence between IO.WLB and IO$_WRITELBLK. The RSX-11M function codes IO.CCO, and IO.WBT are the equivalents of the logical OR of IO.WLB and a subfunction bit. The sections that follow describe VAX/VMS handling of subfunction bits on write-logical-block requests.

5.8.8.1 IO.WLB!TF.CCO and IO.CCO Functions - The RSX-11M function codes IO.WLB!TF.CCO and IO.CCO are equivalent. Either one corresponds directly to the VAX/VMS IO$_WRITELBLK function with a function modifier of IO$M_CANCTRLO.

5.8.8.2 IO.WLB!WBT and IO.WBT Functions - Under VAX/VMS, the write-break-through function is implemented using the Broadcast system service. As a result, neither of the RSX-11M function codes IO.WLB!WBT or IO.WBT corresponds directly to a VAX/VMS driver function. When an RSX-11M image requests write-break-through, VAX/VMS issues a IO$_WRITELBLK function to the driver. A normal write-logical-block function occurs.

5.8.9 IO.WVB Function

The RSX-11M function code IO.WVB corresponds directly to the VAX/VMS function code IO$_WRITEVBLK. VAX/VMS handles the subfunction bits allowed with IO.WVB in the same manner as it handles the subfunction bits for IO.WLB. The resulting I/O operation is a write virtual block, however.

5.8.9.1 IO.WLB!TF.WAL, IO.WAL, and IO.CCO!TF.WAL Functions - The RSX-11M function codes IO.WLB!TF.WAL and IO.WAL are equivalent. The RSX-11M function IO.CCO!TF.WAL adds the cancel CTRL/O subfunction to an IO.WAL request. When an RSX-11M image issues a write-all-data request, VAX/VMS issues an IO$_WRITELBLK!IO$M_NOFORMAT request to cause the data block to be transferred without interpretation to the specified buffer.

VAX/VMS requires an image to have the appropriate privilege to write a physical block. An RSX-11M image must have this privilege to successfully issue an IO.WAL request.

5.8.10 IO.WPB Function

The RSX-11M function code IO.WPB corresponds directly to the VAX/VMS function code IO$_WRITEPBLK. No subfunction bits are applicable.

5.8.11 IO.GTS Function

VAX/VMS has no system generation options that control the features included in the terminal driver.

When an RSX-11M image issues an IO.GTS request, VAX/VMS returns a 4-word buffer of information that describes the VAX/VMS terminal driver features. Because these features cannot be altered, the same information always is returned. Table 5-11 lists the terminal support information returned under VAX/VMS.

That information includes all of the features that can be returned under RSX-11M with the following exceptions, which are always zero:

Word 0,	bit 1	F1.BTW	Write-break-through
	bit 2	F1.BUF	Checkpointing during terminal input
	bit 14	F1.UTP	Input characters buffered in task's address space
	bit 15	F1.VBF	Variable-length terminal buffers

Table 5-11: Information Returned by Get Terminal Support (IO.GTS)

Word	Bit	Mnemonic	Meaning
0	0	F1.ACR	Automatic CRLF on long lines
0	3	F1.UIA	Unsolicited-character-input AST
0	4	F1.CCO	Cancel CTRL/O before writing
0	5	F1.ESQ	Recognize escape sequences in solicited input
0	6	F1.HLD	Hold screen mode
0	7	F1.LWC	Lower-to-uppercase conversion
0	8	F1.RNE	Read with no echo
0	9	F1.RPR	Read after prompting
0	10	F1.RST	Read with special terminators
0	11	F1.RUB	CRT rubout
0	12	F1.SYN	CTRL/R terminal synchronization
0	13	F1.TRW	Read all and write all
1	0	F2.SCH	Set characteristics QUI (SF.SMC)
1	1	F2.GCH	Get characteristics QUI (SF.GMC)
2			Not used in RSX-11M
3			Not used in RSX-11M

NOTE

An IO.GTS function issued for TI, CO, or CL returns no information.

5.8.12 SF.GMC Function

When an RSX-11M image issues an SF.GMC request, VAX/VMS issues an IO$SENSEMODE request. Table 5-12 lists the terminal characteristics that can be returned for SF.GMC requests. The RSX-11M characteristic TC.PRI is never returned by VAX/VMS because VAX/VMS does not incorporate the concept of a privileged terminal.

An SF.GMC function issued for TI, CO, or CL when the device corresponds to a process permanent file becomes a no-op.

5.8.13 SF.SMC Function

When an RSX-11M image issues an SF.SMC function, VAX/VMS issues an IO$_SETMODE request. Table 5-12 provides the correspondence among RSX-11M terminal characteristics bit names and VAX/VMS subfunction modifiers used with the function code IO$_SETMODE.

The set terminal type subfunction (TC.TTP) supports the following terminals: LA36, LA120, LA34, LA38, VT05, VT52, VT55, VT100, VT101, VT102, VT105, VT125, VT131, and VT132. Other terminals are "unknown", including ASR33, ASR35, KSR33, LA30S, LA30P, VT50, VT61, LA180S, LA12, and LA100. The TC.TBF characteristic for the SF.SMC request clears the type-ahead buffer.

An SF.SMC function issued for TI, CO, or CL when the device corresponds to a process-permanent file becomes a no-op.

Table 5-12: Terminal Characteristics for SF.GMC and SF.SMC Requests

RSX-11M Bit Name	VAX/VMS Code	Meaning
TC.BIN	TT$M_PASSALL	No characters are interpreted as control characters (binary mode)
TC.ESQ	TT$M_ESCAPE	Escape-sequence generation
TC.HLD	TT$M_HOLDSCREEN	Hold screen mode
TC.NEC	TT$M_NOECHO	No echo mode
TC.SCP	TT$M_SCOPE	Scope device
TC.SLV	TT$M_NOTYPEAHD	Slave device
TC.SMR	TT$M_LOWER	Lowercase allowed
TC.TBF	--	Number of characters in type-ahead buffer
TC.TTP	Terminal type; see the RSX-11M/M-PLUS I/O Driver's Manual	Terminal type

5.8.14 Terminal Read Status Returns

The contents of an I/O status block used for terminal requests is the same as that used for all QIO operations except for terminal read operations. For terminal read operations, the high-order byte of the first word contains a code that indicates the character or sequence that terminated the read operation. Any one of the following codes can be returned.

Code	Meaning
IS.CR	Read terminated by RETURN
IS.ESC	Read terminated by ALTMODE
IS.ESQ	Read terminated by an escape sequence
--	Other terminator character
0	Read terminated by full buffer

When using VAX/VMS terminal function codes and parameters, keep in mind the following points:

- VAX/VMS terminals can be spooled.
- See Section 3.10 for a discussion of the requirements for issuing IO.WLB and IO.WVB requests to a spooled device.
- If an RSX-11M image issues a GET LUN INFORMATION directive for a spooled device, the information returned is for the intermediate device, that is, for a disk.
- TI, CO, and CL map to VAX/VMS process-permanent files as follows:

RSX-11M Pseudodevice	VAX/VMS Process-Permanent Files
TI	SYS$INPUT and SYS$OUTPUT
CO	SYS$COMMAND
CL	SYS$ERROR

- Process-permanent files are controlled using VAX-11 RMS unless they map to terminals. VAX/VMS, therefore, limits the I/O function codes that can be used to access these files to read and write functions only. All subfunction bits are ignored. Functions other than read and write are illegal and result in the I/O status code IE.IFC (illegal function for this device) being returned.
- For RSX-11M images, user-created process-permanent files appear as record-oriented terminal devices.
- When process-permanent files map to terminals, queue I/O requests can be issued.
- The device characteristics for TI, CO, and CL are as follows:

  Unit record device
  Terminal
  132-byte buffer
  Carriage control
  No lowercase

5.9 CARD READER DRIVER

Table 5-13 provides the correspondence between RSX-11M card reader functions and VAX/VMS function codes or resultant actions.

Table 5-13: Card Reader Function Code Correspondence

Function	RSX-11M Code	VAX/VMS Code or Action
Attach Device	IO.ATT	No operation
Detach Device	IO.DET	No operation
Cancel I/O Request	IO.KIL	Cancel I/O on Channel system service
Read Virtual Block	IO.RVB	IO$_READVBLK
Read Logical Block	IO.RLB	IO$_READLBLK
Read Logical Block	IO.RBD	IO$_READLBLK!IO$M_BINARY

The two function-dependent parameters (P1 and P2) for RSX-11M card reader functions correspond directly to P1 and P2 of VAX/VMS card reader functions.

5.10 NULL DEVICE

VAX/VMS supports the use of a null device by RSX-11M images. As under RSX-11M, a read request to the null device results in an end-of-file status return (IE.EOF), and a write request results in success status return (IE.SUC).

I/O to the null device is treated like I/O to an unsupported device as described in Section 5.1.

5.11 DISK AND MAGNETIC TAPE ACPS

I/O operations involving file-structured devices (disk and magnetic tape) often require ACP intervention. Normally, RSX-11M images perform I/O using RMS-11 or FCS; they do not issue QUEUE I/O REQUEST directives directly to an ACP. Any ACP intervention needed is requested by RMS-11 or FCS and occurs transparently from the image's point of view. It is possible, however, for images to request ACP functions directly by issuing a QUEUE I/O REQUEST directive and specifying an ACP function code.

The information in this section is relevant only to RSX-11M images that issue ACP functions directly, for example, create file and enter file name. Other RSX-11M images running under VAX/VMS can rely on RMS-11 or FCS to request appropriate RSX-11M ACP functions during image execution.

VAX/VMS ACP functions are expressed using six function codes and three function modifiers. The six function codes follow:

1. IO$_CREATE -- Create file
2. IO$_ACCESS -- Access file
3. IO$_DEACCESS -- Deaccess file
4. IO$_MODIFY -- Modify file
5. IO$_DELETE -- Delete file
6. IO$_ACPCONTROL -- ACP control

The three function modifiers, which can be applied to the create, access, and delete functions, follow:

1. IO$M_ACCESS -- Open file on user's channel
2. IO$M_CREATE -- Create a file identification
3. IO$M_DELETE -- Delete file

By using a function code and a function modifier together, an image can request multiple ACP operations in one I/O request. For example, IO$_CREATE!IO$M_ACCESS requests the ACP to create a file and to access the file on the specified channel. IO$_DELETE!IO$M_DELETE causes a file's directory entry and file header to be deleted; that is, the file is deleted. IO$_DELETE with no function modifier causes the file's directory entry to be deleted.

In addition to function codes and modifiers, VAX/VMS ACPs use a file identification block (FIB) for communication between the requester and the ACP. The function-dependent parameter P1 for all ACP requests is the address of a descriptor for the associated FIB. The FIB contains much of the information passed to an ACP by an RSX-11M image in P1 through P6. Figure 5-2 illustrates a FIB. The VAX/VMS I/O User's Guide provides a detailed description of the contents of a FIB and describes the ACP functions supported by VAX/VMS.

RSX-11M ACP functions are expressed using the following function codes:

- IO.CRE -- Create file
- IO.ACR -- Access for read
- IO.ACW -- Access for write
- IO.ACE -- Access for extend
- IO.EXT -- Extend file
- IO.WAT -- Write attributes
- IO.RAT -- Read attributes
- IO.DAC -- Deaccess file
- IO.DEL -- Delete file

- IO.FNA -- Find file name
- IO.RNA -- Remove file name
- IO.ENA -- Enter file name
- IO.APC -- ACP control

When an RSX-11M image issues an ACP request under VAX/VMS, VAX/VMS issues a Queue I/O Request system service to the ACP. It obtains the data to fill in the FIB and function-dependent parameters for the request from two sources:

- Function-dependent parameters supplied by the image in the QUEUE I/O REQUEST directive
- Data structures pointed to by function-dependent parameters, for example, the file name block

Once the requested function is performed, VAX/VMS fills in the RSX-11M image's data structures with the same information that is returned to the image when executing under the RSX-11M operating system.

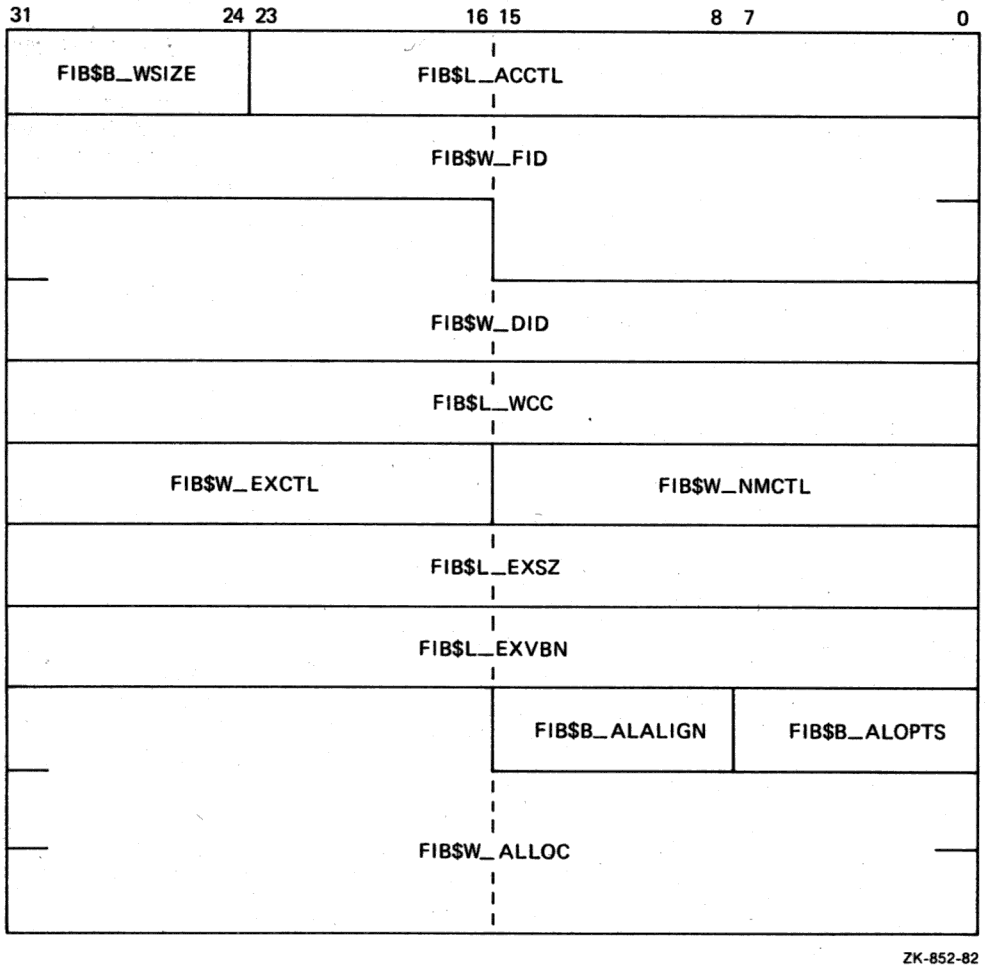


Figure 5-2: File Identification Block Format

5.11.1 General Correspondence of Parameters

Table 5-14 identifies the relationship of RSX-11M function-dependent parameters to VAX/VMS function-dependent parameters and FIB fields.

Table 5-14: ACP Parameter Correspondence

Parameter Function	RSX-11M Pn	VAX/VMS Equivalent
File identification pointer	P1 (pointer)	FIB$W_FID (value)
Attribute list pointer	P2	P5 (reformatted)
Extend control	P3 (high byte)	FIB$W_EXCTL
Delta size in blocks	P3 (low byte) and P4	FIB$L_EXVBN for truncate only; FIB$L_EXSZ for extend
Window size	P5 (low byte)	FIB$B_WSIZE
Access control	P5 (high byte)	FIB$L_ACCTL
File name block pointer	P6	P2 (name string) and P4 (result string)

5.11.2 IO.CRE Function

Equivalent Function Code: IO$_CREATE!IO$M_CREATE

Notes:

1. If the extend size is supplied in the low-order byte of P3 and in P4, it is stored in FIB$L_EXSZ.

2. The high-order byte of P3 (extend control) is used to set bits in FIB$W_EXCTL:

```
FIB$V_EXTEND  = EX.ENA
FIB$V_ALCON   = EX.AC1
FIB$V_ALCONB  = EX.AC2
FIB$V_FILCON  = EX.FCO
FIB$V_ALDEF   = EX.ADF
```

3. The file identification is copied from FIB$W_FID and returned in the address pointed to by P1 (FID pointer).

4. Information in the VAX/VMS attribute list is derived from the RSX-11M attribute list, if one is supplied.

5. The extend size in blocks is returned in bytes 1, 2, and 3 of the I/O status block.

5.11.3 IO.DEL with EX.ENA=0

Equivalent Function Code: IO$_DELETE!IO$M_DELETE

Note:

1. The file identification pointed to by P1 is copied into FIB$W_FID.

5.11.4 IO.DEL with EX.ENA=1

Equivalent Function Code: IO$_MODIFY

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.
2. FIB$V_TRUNC is set in field FIB$W_EXCTL.
3. The extend size supplied in the low byte of P3 and in P4 is incremented by 1 and stored in FIB$L_EXVBN.
4. The file round-up in blocks is returned in bytes 2 and 3 of the I/O status block. File round-up is the number of blocks added to the specified file size to reach the next cluster boundary.

5.11.5 IO.ACR Function

Equivalent Function Code: IO$_ACCESS!IO$M_ACCESS

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.
2. The high-order byte of P5 (access control) is used to set bits in FIB$L_ACCTL:

```
FIB$V_NOWRITE  = AC.LCK
FIB$V_REWIND   = AC.RWD
FIB$V_CURPOS   = AC.POS
FIB$V_UPDATE   = AC.UPD
```

3. The window size provided by the low-order byte of P5 is stored in FIB$B_WSIZE.
4. Information in the VAX/VMS attribute list is derived from the RSX-11M attribute list, if one is supplied.

5.11.6 IO.ACW and IO.ACE Functions

Equivalent Function Code: IO$_ACCESS!IO$M_ACCESS

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.

2. The high-order byte of P5 (access control) is used to set bits in FIB$L_ACCTL:

```
FIB$V_DLOCK   = AC.DLK
FIB$V_NOWRITE = AC.LCK
FIB$V_REWIND  = AC.RWD
FIB$V_CURPOS  = AC.POS
FIB$V_UPDATE  = AC.UPD
```

   In addition, VAX/VMS sets FIB$V_WRITE.

3. The window size provided by the low-order byte of P5 is stored in FIB$B_WSIZE.

4. Information in the VAX/VMS attribute list is derived from the RSX-11M attribute list, if one is supplied.

5.11.7 IO.DAC Function

Equivalent Function Code: IO$_DEACCESS

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.

2. Information in the VAX/VMS attribute list is derived from the RSX-11M attribute list, if one is supplied.

5.11.8 IO.EXT Function

Equivalent Function Code: IO$_MODIFY

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.

2. The high-order byte of P3 (extend control) is used to set bits in FIB$W_EXCTL:

```
FIB$V_EXTEND  = EX.ENA
FIB$V_ALCON   = EX.AC1
FIB$V_ALCONB  = EX.AC2
FIB$V_FILCON  = EX.FCO
FIB$V_ALDEF   = EX.ADF
```

3. The extend size supplied in the low-order byte of P3 and in P4 is stored in FIB$L_EXSZ.

4. The amount by which the file is extended is returned in bytes 1, 2, and 3 of the I/O status block.

5.11.9 IO.WAT Function

Equivalent Function Code: IO$_MODIFY

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.

2. Information in the VAX/VMS attribute list is derived from the RSX-11M attribute list, if one is supplied.

5.11.10 IO.RAT Function

Equivalent Function Code: IO$_ACCESS

Notes:

1. The file identification pointed to by P1 is copied into FIB$W_FID.

2. Information in the VAX/VMS attribute list is derived from the RSX-11M attribute list, if one is supplied.

5.11.11 IO.FNA Function

Equivalent Function Code: IO$_ACCESS

Notes:

1. The file identification is copied from FIB$W_FID and returned in the address pointed to by P1.

2. The directory identification is copied from the file name block into FIB$W_DID.

3. The file name string supplied in the request is constructed from the Radix-50 file name in the file name block.

4. If the bit NB.WLV is set in N.STAT of the file name block, a resultant string is constructed from the Radix-50 name and type. The version number is stored in N.FID+4 of the name block, and is supplied as input to the IO$_ACCESS call.

   If NB.WLV is not set, a resultant string of zero length is supplied.

5. The file name string returned is the resultant string returned by the Queue I/O Request system service. It is converted back to Radix-50 and returned to the file name block.

6. Control bits in field N.STAT of the file name block are used to set bits of FIB$W_NMCTL:

```
FIB$V_ALLNAM  = NB.SNM
FIB$V_ALLTYP  = NB.STP
FIB$V_ALLVER  = NB.SVR
FIB$V_WILD    = NB.SNM!NB.STP!NB.SVR
```

7. The file name block field N.NEXT is used to set FIB$L_WCC. The resulting value of FIB$L_WCC is returned in N.NEXT.

5.11.12 IO.RNA Function

Equivalent Function Code: IO$_DELETE

Notes:

1. The file identification is copied from FIB$W_FID and returned in the address pointed to by P1.

2. The directory identification is copied from the file name block into FIB$W_DID.

3. If the bit NB.WLV is set in N.STAT of the file name block, a resultant string is constructed from the Radix-50 name and type. The version number is stored in N.FID+4 of the name block, and is supplied as input to the IO$_ACCESS call.

   If NB.WLV is not set, a resultant string of zero length is supplied.

4. The file name string supplied in the request is constructed from the Radix-50 file name in the file name block.

5. The file name returned is the resultant string returned by the Queue I/O Request system service. It is converted back to Radix-50 and returned to the file name block.

6. Control bits in field N.STAT of the file name block are used to set bits of FIB$W_NMCTL:

```
FIB$V_ALLNAM  = NB.SNM
FIB$V_ALLTYP  = NB.STP
FIB$V_ALLVER  = NB.SVR
FIB$V_WILD    = NB.SNM!NB.STP!NB.SVR
```

7. The file name block field N.NEXT is used to set FIB$L_WCC. The resulting value of FIB$L_WCC is returned in N.NEXT.

5.11.13 IO.ENA Function

Equivalent Function Code: IO$_CREATE

Notes:

1. The file identification is copied from the file name block into FIB$W_FID.

2. The directory identification is copied from the file name block into FIB$W_DID.

3. The file name string supplied in the request is constructed from the Radix-50 file name in the file name block.

4. The file name returned is the resultant string returned by the Queue I/O Request system service. It is converted back to Radix-50 and returned to the file name block.

5.11.14 IO.APC Function

Equivalent Function Code: IO$_ACPCONTROL

Notes:

1. P3 contains the subfunction identification. The low-order byte of P3 is zero-extended and stored at FIB$W_CNTRLFUNC, which overlays FIB$W_EXCTL. The RSX-11M ACP subfunction codes have direct equivalents in VAX/VMS, as follows.

RSX-11M Subfunction	VAX/VMS Subfunction
FF.NV	FIB$C_NEXTVOL
FF.POE	FIB$C_POSEND
FF.RWD	FIB$C_REWINDVOL
FF.RWF	FIB$C_REWINDFIL
FF.SPC	FIB$C_SPACE

2. For the FF.SPC subfunction, P4 is sign-extended and stored at FIB$L_CNTRLVAL, which overlays FIB$L_EXSZ. A negative value for P4 specifies the number of blocks to space backward. A positive value indicates the number of blocks to space forward.

APPENDIX A

VAX-11 COMPATIBILITY MODE INSTRUCTION SET

Table A-1 lists the VAX-11 compatibility mode instruction set.

Table A-1: VAX-11 Compatibility Mode Instruction Set

Opcode (octal)	Mnemonic
000002	RTI
000006	RTT
0001DD	JMP
00020R	RTS
000240-000277	Condition codes
0003DD	SWAB
000400-003777	Branches
100000-103777	Branches
004RDD	JSR
.050DD	CLR(B)
.051DD	COM(B)
.052DD	INC(B)
.053DD	DEC(B)
.054DD	NEG(B)
.055DD	ADC(B)
.056DD	SBC(B)
.057DD	TST(B)
.060DD	RCR(B)
.061DD	ROL(B)
.062DD	ASR(B)
.063DD	ASL(B)
0065SS	MFPI (See note below)
0066DD	MTPI (See note below)
1065SS	MFPD (See note below)
1066DD	MTPD (See note below)
0067DD	SXT
070RSS	MUL
071RSS	DIV
072RSS	ASH
073RSS	ASHC
074RSS	XOR
077RNN	SOB
.1SSDD	MOV(B)
.2SSDD	CMP(B)
.3SSDD	BIT(B)
.4SSDD	BIC(B)
.5SSDD	BIS(B)
06SSDD	ADD
16SSDD	SUB

NOTE

The MFPI, MTPI, MFPD, and MTPD, instructions execute exactly as they would on a PDP-11 in user mode with Instruction and Data space overmapped. More specifically, they ignore the previous access level and act like PUSH and POP instructions referring to the current stack.

VAX/VMS provides emulation of the FPP floating-point instructions.

APPENDIX B

PARSE DIRECTIVE

The parse directive allows an RSX-11M image to use VAX/VMS file specifications that are not fully qualified because of the use of logical names. Use of this directive replaces the operation of the FCS .PARSE, .PRSDR, and .PRSDV routines and the RMS-11 $PARSE routine for RSX-11M images running in VAX/VMS.

An RSX-11M image requests the parsing of a file specification by issuing a parse directive that supplies the addresses of a file name block and data structures containing default information. VAX/VMS uses the information supplied by the image and information contained in the RSX-11M logical name table and the system logical name table to build the primary and default strings that VAX-11 RMS requires to perform the actual parsing. VAX-11 RMS returns the expanded name to VAX/VMS. VAX/VMS, in turn, uses the expanded name to fill in the appropriate RSX-11M data structures, for example, returned directory string and file name block. The result is that the image receives the information in the normal RSX-11M formats.

The image can request four different types of parsing:

1. Parsing of the full file specification (normal mode)
2. Parsing of the device name only (device-only mode)
3. Parsing of the file name using the default file name block as the major source of input (dfnb mode)
4. RMS-11 mode of parsing

B.1 NORMAL MODE PARSING

When the mode parameter is equal to 0, VAX-11 RMS parses the full file specification. VAX/VMS builds the primary string required as input to VAX-11 RMS by concatenating fields of the dataset descriptor, as follows:

- Device
- Directory
- Filename.type;version

It builds the default string from fields of the default file name block and from the default directory descriptor, as follows:

- Device from the LUN or default file name block
- Default directory from the image's default directory descriptor
- Filename.type;version from the default file name block

VAX-11 RMS returns to the RSX-11M image a filled-in file name block and directory string descriptor in the file name block. The directory string is returned at the address specified in the descriptor.

B.2 DEVICE-ONLY PARSING

When the mode parameter is equal to 1, VAX-11 RMS parses only the device and directory portion of the file specification. It uses the same sources for the primary and default strings as it does for a normal parsing operation.

B.3 DEFAULT FILENAME BLOCK PARSING

When the mode parameter is equal to 2, VAX/VMS uses the Radix-50 file name in the default file name block to build the ASCII file name for the primary string.

For the default string, VAX/VMS takes the device name from the default file name block. It takes the directory name from the default directory descriptor, and the file name, type, and version from the default file name block.

The DSW return codes for default file name block parsing are the same as for normal mode parsing.

B.4 RMS-11 PARSING

When the mode parameter is equal to 3, VAX-11 RMS parses the file specification using the same method used by RMS-11. The format for the DPB is slightly different from that used for modes 0, 1, and 2, as described below.

B.5 DIRECTIVE CALL AND DPB FORMATS

The parse directive is called using DIR$, as follows:

```
DIR$   #pardpb
```

The DPB has the following format for modes 0, 1, and 2.

```
pardpb:   .BYTE  145.,7
          .WORD  mode
          .WORD  lun
          .WORD  dspt
          .WORD  dfnb
          .WORD  dfdd
          .WORD  fnb
          .WORD  rtdd
```

mode = 0 for normal, 1 for device-only, 2 for default file name block, or 3 for RMS-11. See the sections that follow for a description.

lun = logical unit number.

dspt = address of the data set descriptor.

dfnb = address of the default name block.

dfdd = address of the descriptor for the default directory string. See the first note below.

fnb = address of the file name block.

rtdd = address of the descriptor for the returned directory string. See the first note below.

The DPB has the following format for mode 3.

```
pardpb:     .BYTE   145.,7
            .WORD   mode
            .WORD   lun
            .WORD   pript
            .WORD   did
            .WORD   0 (not used)
            .WORD   fnb
            .WORD   expnam
```

The definitions of mode, lun, and fnb are the same as those for the DPB format provided above.

pript = address of the primary input descriptor.

did = address of the default input descriptor.

expnam = address of the descriptor for the block in which to return the expanded name.

DSW Return Codes:

```
IS.SUC  --  Success
IE.BAD  --  Invalid mode missing or bad parameter (default error)
IE.NSF  --  Directory not found (RMS$_DNF)
IE.BDI  --  Bad directory syntax (RMS$_DIR)
IE.BNM  --  Bad file name (RMS$_(SYN,FNM,LNE,TYP,VER))
IE.DNR  --  Device not ready (RMS$_DNR)
IE.DUN  --  Device not available (RMS$_CHN)
IE.NSF  --  File not found (RMS$_FNF)
IE.BDV  --  Bad device specification (RMS$_DEV)
```

Notes:

1. All descriptor input parameters must be a 2-word block with the following format.

```
.WORD   size
.WORD   address
```

2. RSX-11M does not support the parse directive. An RSX-11M image using this directive can test for an illegal directive DSW code to determine whether it is executing under RSX-11M or VAX/VMS and take appropriate action at run time.

INDEX

ABORT TASK directive, 4-8
ABRT$ macro, 4-8
Account, 2-2
ACP (Ancillary Control Process),
  disk, 5-23
  Files-11 mapping, 3-11
  I/O, 3-4
  IO.ACE function, 5-28
  IO.ACR function, 5-27
  IO.ACW function, 5-28
  IO.APC function, 5-31
  IO.CRE function, 5-26
  IO.DAC function, 5-28
  IO.DEL function, 5-27
  IO.ENA function, 5-30
  IO.EXT function, 5-28
  IO.FNA function, 5-29
  IO.RAT function, 5-29
  IO.RNA function, 5-30
  IO.WAT function, 5-29
  magnetic tape, 5-23
  parameter correspondence, 5-26
  relationship with VAX/VMS I/O system components, 3-1
  requested by FCS, 5-23
  requested by RMS-11, 5-23
ALLOCATE command, 5-2, 5-11
Allocate Device ($ALLOC) system service, 3-4, 5-2
Allocation,
  implicit, 5-2
ALTER PRIORITY directive, 4-9
ALTP$ macro, 4-9
ALUN$ macro, 4-10
Ancillary Control Process (ACP),
  See ACP (Ancillary Control Process)
ASN command, 3-9
Assign device, 3-5, 4-10
Assign I/O Channel ($ASSIGN) system service, 3-3, 4-10
ASSIGN LUN directive, 3-5, 4-10
Associate Common Event Flag Cluster ($ASCEFC) system service, 2-4, 4-14
AST (Asynchronous System Trap),
  declare, 4-33, 4-35, 4-37
  disable recognition, 4-17
  enable recognition, 4-20
  floating-point, 4-51
AST (Asynchronous System (Cont.)
  power recovery, 4-53
  receive data, 4-55
  service routine, 2-10
  service routine termination, 4-11
  termination, 4-54
AST SERVICE EXIT directive, 4-11
ASTX$S macro, 4-11
Asynchronous System Trap (AST),
  See AST (Asynchronous System Trap)
ATRG$ macro, 4-2
ATTACH REGION directive, 4-2

Balance set, 2-6
Block locking, 3-5

Cancel I/O on Channel ($CANCEL) system service, 3-4, 5-3, 5-23
CANCEL MARK TIME REQUESTS directive, 4-13
CANCEL TIME BASED INITIATION REQUESTS directive, 4-15
Cancel Timer Request ($CANTIM) system service, 4-13
Cancel Wakeup ($CANWAK) system service, 4-15
Card reader driver, 5-23
Channel,
  I/O assignment, 3-3, 4-10
Checkpointing,
  disable, 4-18
  enable, 4-21
CINT$ macro, 4-2
CL pseudodevice, 3-8
Clear Event Flag ($CLREF) system service, 4-12
CLEAR EVENT FLAG directive, 4-12
CLEF$ macro, 4-12
Clock,
  directive, 2-8
  system, 2-8, 4-31
CMKT$ macro, 4-13
CNCT$ macro, 4-2
CO pseudodevice, 3-8

Code,
  DSW, 2-5, 4-7
  I/O status, 5-3
  system status, 4-7, 4-24
  termination, 2-10
Command,
  ALLOCATE, 5-2, 5-11
  ASN, 3-9
  RUN, 2-2, 4-29
  SET TERMINAL, 5-16
Command line,
  MCR, 4-28
Common area, 2-9
Common event flag, 2-4
Common option, 2-9
Communication,
  interprocess, 2-7, 3-10
Compatibility,
  FCS, 3-5
  instruction set, A-1
  mode, 1-1
  PDP-11, 1-2
  RMS-11, 3-5
  RSX-11M, 1-2
Concealed device, 3-8
CONNECT directive, 4-2
CONNECT TO INTERRUPT VECTOR directive, 4-2
Control region, 1-5
Conversion,
  event flag, 2-4
  I/O code, 5-3
  physical device name, 3-7
  pseudodevice name, 3-8
CRAW$ macro, 4-2
CREATE ADDRESS WINDOW directive, 4-2
CREATE GROUP GLOBAL EVENT FLAGS directive, 4-14
Create Mailbox and Assign I/O Channel ($CREMBX) system service, 3-3
CREATE REGION directive, 4-2
CRGF$ macro, 4-14
CRRG$ macro, 4-2
CSRQ$ macro, 4-15

Debug,
  SST vector table for, 4-60
DECL$S macro, 4-16
DECLARE SIGNIFICANT EVENT directive, 2-7, 4-16
DETACH REGION directive, 4-2
Detached process, 2-10
Device,
  See also Pseudodevice
  allocate, 3-4, 5-2
Device (Cont.)
  assignment, 3-5, 4-10
  buffer size information, 4-26
  cancel I/O to, 5-3
  card reader, 5-23
  characteristics, 3-4, 4-26
  concealed, 3-8
  deallocate, 3-4, 5-2
  disk, 5-8
  file structured, 5-23
  information, 4-26, 5-2
  line printer, 5-10
  logical name, 3-7 to 3-9, 4-10
  magnetic tape, 5-9
  mapping name between VAX/VMS and RSX-11M, 3-7 to 3-8
  name, 3-7 to 3-9, 4-10
  nonshareable, 3-4, 5-2
  null, 5-23
  physical, 4-10
  queue, 4-35, 4-37
  shareable, 3-4, 5-2
  spooled, 3-11 to 3-12, 5-11, 5-22
  supported, 5-2
  SYS$COMMAND, 3-8
  SYS$DISK, 3-8
  SYS$ERROR, 2-11, 3-8
  SYS$INPUT, 3-8
  SYS$OUTPUT, 2-11, 3-8
  SYS$SCRATCH, 3-8
  SYS$SYSROOT, 3-8 to 3-9
  terminal, 5-11
  unit number, 3-7
  unknown terminal, 5-21
DIR$ macro, B-1
Directive,
  ABORT TASK, 4-8
  added since RSX-11M Version 3.2, 4-1
  ALTER PRIORITY, 4-9
  ASSIGN LUN, 3-5, 4-10
  AST SERVICE EXIT, 4-11
  ATTACH REGION, 4-2
  CANCEL MARK TIME REQUESTS, 4-13
  CANCEL TIME BASED INITIATION REQUESTS, 4-15
  CLEAR EVENT FLAG, 4-12
  compatibility, 1-3
  CONNECT, 4-2
  CONNECT TO INTERRUPT VECTOR, 4-2
  CREATE ADDRESS WINDOW, 4-2
  CREATE GROUP GLOBAL EVENT FLAGS, 4-14
  CREATE REGION, 4-2

Directive (Cont.)
DECLARE SIGNIFICANT EVENT, 2-7, 4-16
DETACH REGION, 4-2
DISABLE AST RECOGNITION, 4-17
DISABLE CHECKPOINTING, 2-6 to 2-7, 4-18
ELIMINATE ADDRESS WINDOW, 4-3
ELIMINATE GROUP GLOBAL EVENT FLAGS, 4-19
emulation, 1-3
ENABLE AST RECOGNITION, 4-20
ENABLE CHECKPOINTING, 2-6, 4-21
EXIT IF, 4-22
EXIT WITH STATUS, 4-24
EXTEND TASK, 4-25
GET LUN INFORMATION, 3-4, 3-12, 4-26, 5-2
GET MAPPING CONTEXT, 4-3
GET MCR COMMAND LINE, 4-28
GET PARTITION PARAMETERS, 2-7, 4-30
GET REGION PARAMETERS, 4-3
GET SENSE SWITCHES, 4-3
GET TASK PARAMETERS, 4-32
GET TIME PARAMETERS, 2-8, 4-31
INHIBIT AST RECOGNITION, 4-17
MAP ADDRESS WINDOW, 4-4
MARK TIME, 2-8, 4-33
PLAS (program logical address space), 1-4
QUEUE I/O REQUEST, 3-12, 4-35, 5-1, 5-3
QUEUE I/O REQUEST AND WAIT, 4-37
READ ALL EVENT FLAGS, 4-42
READ EXTENDED EVENT FLAGS, 4-43
RECEIVE BY DIFFERENCE, 4-5
RECEIVE DATA, 3-10, 4-40
RECEIVE DATA OR EXIT, 3-10, 4-41
RECEIVE DATA OR STOP, 3-10, 4-38
REQUEST TASK, 2-10, 4-44
RESUME TASK, 2-10, 4-45
RUN TASK, 2-10, 4-46
SEND BY REFERENCE, 4-5
SEND DATA, 3-10, 3-12, 4-48
SET EVENT FLAG, 4-50
SPAWN, 4-54
SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST, 4-51

Directive (Cont.)
SPECIFY POWER RECOVERY AST, 4-53
SPECIFY RECEIVE DATA AST, 4-55
SPECIFY RECEIVE-BY-REFERENCE AST, 4-5
SPECIFY SST VECTOR TABLE FOR DEBUGGING AID, 4-60
SPECIFY SST VECTOR TABLE FOR TASK, 4-61
STOP, 4-58
STOP FOR LOGICAL OR OF EVENT FLAGS, 4-57
STOP FOR SINGLE EVENT FLAG, 4-59
summary of, 4-1 to 4-6
SUSPEND, 4-52
TASK EXIT, 4-23
UNMAP ADDRESS WINDOW, 4-6
UNSTOP TASK, 2-10, 4-62
unsupported, 4-1 to 4-5
WAIT FOR LOGICAL OR EVENT FLAGS, 4-65
WAIT FOR SIGNIFICANT EVENT, 2-7, 4-64
WAIT FOR SINGLE EVENT FLAG, 4-66
Directory,
MFD (Master File Directory), 3-9
root, 3-9
DISABLE AST RECOGNITION directive, 4-17
DISABLE CHECKPOINTING directive, 2-6 to 2-7, 4-18
Disassociate Common Event Flag Cluster ($DACEFC) system service, 4-19
Disk,
ACP, 5-23
driver, 5-8
Dismount ($DISMOU) system service, 3-4
Driver,
card reader, 5-23
disk, 5-8
I/O, 5-1
line printer, 5-10
magnetic tape, 5-9
terminal, 5-11
DSAR$S macro, 4-17
DSCP$S macro, 4-18
DSW (Directive Status Word) code, 2-5 to 2-6, 4-7
DTRG$ macro, 4-2

ELAW$ macro, 4-3
ELGF$ macro, 4-19
ELIMINATE ADDRESS WINDOW directive, 4-3
ELIMINATE GROUP GLOBAL EVENT FLAGS directive, 4-19
Emulation,
  directive, 1-3
  floating-point, 1-1 to 1-2, 1-4
  I/O, 5-1
ENABLE AST RECOGNITION directive, 4-20
ENABLE CHECKPOINTING directive, 2-6, 4-21
ENAR$S macro, 4-20
ENCP$S macro, 4-21
Error condition, 2-10 to 2-11
Event,
  significant, 2-7, 4-16
  system, 2-7
Event flag,
  clear, 4-12
  cluster, 2-4
  common, 2-4, 4-42 to 4-43
  conversion, 2-4
  extended, 4-43
  group global, 2-4, 4-14, 4-19, 4-43
  I/O related, 4-35, 4-37
  local, 2-4, 4-42 to 4-43
  logical OR, 4-57, 4-64 to 4-65
  protection, 2-4
  read, 4-42 to 4-43
  RSXCOMEFN cluster, 2-4
  RSXGROUPEFN cluster, 2-5
  set, 4-35, 4-48, 4-50
  single, 4-59, 4-66
  stop for logical OR, 4-57
  stop for single, 4-59
  wait for logical OR, 4-65
  wait for single, 4-66
Event-associated directive,
  CANCEL MARK TIME REQUESTS, 4-13
  CLEAR EVENT FLAG, 4-12
  CREATE GROUP GLOBAL EVENT FLAGS, 4-14
  DECLARE SIGNIFICANT EVENT, 4-16
  ELIMINATE GROUP GLOBAL EVENT FLAGS, 4-19
  EXIT IF, 4-22
  MARK TIME, 4-33
  READ ALL EVENT FLAGS, 4-42
  READ EXTENDED EVENT FLAGS, 4-43
  SET EVENT FLAG, 4-50
Event-associated directive (Cont.)
  STOP FOR LOGICAL OR OF EVENT FLAGS, 4-57
  STOP FOR SINGLE EVENT FLAG, 4-59
  WAIT FOR LOGICAL OR OF EVENT FLAGS, 4-65
  WAIT FOR SIGNIFICANT EVENT, 4-64
  WAIT FOR SINGLE EVENT FLAG, 4-66
EXIF$ macro, 4-22
Exit ($EXIT) system service, 2-11, 4-22 to 4-24, 4-41
EXIT IF directive, 4-22
EXIT WITH STATUS directive, 4-24
EXIT$S macro, 4-23
EXST$ macro, 4-24
EXTEND TASK directive, 4-25
Extended event flag, 4-43
EXTK$ macro, 4-25

FCS (File Control System), 3-5
  spool function, 3-12
File identification block (FIB), 5-24 to 5-25
File specification,
  negative version number, 2-13
  parse, 2-13
  parse directive, B-1
File-structured device, 5-23
Files-11 ACP, 3-11
Floating-point,
  AST, 4-51
  emulation, 1-1 to 1-2, 1-4
  instruction, 1-1, 1-4
  performance, 1-4
Force Exit ($FORCEX) system service, 4-8

Get Device/Volume Information ($GETDVI) system service, 3-4
Get I/O Channel Information ($GETCHN) system service, 4-27
GET LUN INFORMATION directive, 3-4, 3-12, 4-26, 5-2
GET MAPPING CONTEXT directive, 4-3
GET MCR COMMAND LINE directive, 4-28

GET PARTITION PARAMETERS
directive, 2-7, 4-30
GET REGION PARAMETERS
directive, 4-3
GET SENSE SWITCHES directive,
4-3
GET TASK PARAMETERS directive,
4-32
Get Time ($GETTIM) system
service, 4-31
GET TIME PARAMETERS directive,
2-8, 4-31
Global section, 2-8
GLUN$ macro, 4-26
GMCR$ macro, 4-28
GMCX$ macro, 4-3
GPRT$ macro, 4-30
GREG$ macro, 4-3
Group,
UIC, 2-1
Group global event flag, 2-4,
4-14, 4-19
GSSW$S macro, 4-3
GTIM$ macro, 4-31
GTSK$ macro, 4-32

HALT instruction, 1-2
Hibernate ($HIBER) system
service, 4-38 to 4-39,
4-52, 4-58
Hibernation, 2-10

I/O,
ACP, 3-4
block locking, 3-5
cancel, 3-4, 5-3
card reader, 5-23
channel, 3-3, 4-10
code conversion, 5-3
directive differences, 2-13
disk, 5-8
driver, 3-4, 5-1
See also Driver
emulation, 5-1
function code, 5-1 to 5-2
interface to VAX/VMS, 3-5
limits on resource usage,
2-2
line printer, 5-10
magnetic tape, 5-9
mailbox, 3-10
null device, 5-23
relationship with system
components, 3-1
request, 3-3, 3-9
I/O (Cont.)
resource usage limits, 2-2
return, 5-3
standard functions, 5-2
status, 4-35, 4-37
status block, 5-3
system, 2-13, 3-1
system services for, 3-2
terminal, 5-11
I/O and interprocess
communication related
directive,
ASSIGN LUN, 4-10
GET LUN INFORMATION, 4-26
GET MCR COMMAND LINE, 4-28
QUEUE I/O REQUEST, 4-35
QUEUE I/O REQUEST AND WAIT,
4-37
RECEIVE DATA, 4-40
RECEIVE DATA OR EXIT, 4-41
SEND DATA, 4-48
Identification code,
user, 2-1
IHAR$S macro, 4-17
Image,
interface to VAX/VMS I/O,
3-4
multiuser task, 1-4
shareable, 2-7
termination, 2-10 to 2-11,
4-8, 4-22 to 4-24, 4-41
VAX/VMS, 1-5
virtual address location,
1-5
Implicit allocation, 5-2
Information,
device, 4-26, 5-2
event flag, 4-42
logical unit number, 4-26,
4-32
mailbox, 4-26
partition, 4-30, 4-32
priority, 4-32
process, 4-32
SST vector table, 4-32
task image, 4-32
time parameters, 4-31
UIC, 2-2, 4-32
Informational directive,
GET PARTITION PARAMETERS,
4-30
GET TASK PARAMETERS, 4-32
GET TIME PARAMETERS, 4-31
INHIBIT AST RECOGNITION
directive, 4-17
Installed global section, 2-9
Installed task, 2-3
Instruction,
compatibility, 1-1

Instruction (Cont.)
EAE, 1-1
EMT 377, 1-1
FIS, 1-1
floating-point, 1-1, 1-4
FPP, 1-1, 1-4
HALT, 1-2
RESET, 1-2
Instruction set,
compatibility mode, A-1
PDP-11, 1-2
VAX-11, 1-1
Interprocess communication,
2-7, 3-10
See also I/O and interprocess communication related directive

LB pseudodevice, 3-8 to 3-9
LIBR option, 2-9
Library, 2-9
Limits,
resource usage, 2-2
Line printer driver, 5-10
Local event flag, 2-4, 4-43
Logical name,
device, 3-7, 4-10
system-defined, 3-8
translation, 3-9
Logical unit,
information, 4-26, 4-32

Macro,
ABRT$, 4-8
ALTP$, 4-9
ALUN$, 4-10
ASTX$S, 4-11
ATRG$, 4-2
CINT$, 4-2
CLEF$, 4-12
CMKT$, 4-13
CNCT$, 4-2
CRAW$, 4-2
CRGF$, 4-14
CRRG$, 4-2
CSRQ$, 4-15
DECL$S, 4-16
DIR$, B-1
DSAR$S, 4-17
DSCP$S, 4-18
DTRG$, 4-2
ELAW$, 4-3
ELGF$, 4-19
ENAR$S, 4-20

Macro (Cont.)
ENCP$S, 4-21
EXIF$, 4-22
EXIT$S, 4-23
EXST$, 4-24
EXTK$, 4-25
GLUN$, 4-26
GMCR$, 4-28
GMCX$, 4-3
GPRT$, 4-30
GREG$, 4-3
GSSW$S, 4-3
GTIM$, 4-31
GTSK$, 4-32
IHAR$S, 4-17
MAP$, 4-4
MRKT$, 4-33
PRINT$, 3-12
QIO$, 4-35
QIOW$, 4-37
RCST$, 4-38
RCVD$, 4-40
RCVX$, 4-41
RDAF$, 4-42
RDXF$, 4-43
RQST$, 4-44
RREF$, 4-5
RSUM$, 4-45
$RSXDEF, 2-11
RUN$, 4-46
SDAT$, 4-48
SETF$, 4-50
SFPA$, 4-51
SPND$S, 4-52
SPRA$, 4-53
SPWN$, 4-54
SRDA$, 4-55
SREF$, 4-5
SRRA$, 4-5
STLO$, 4-57
STOP$S, 4-58
STSE$, 4-59
SVDB$, 4-60
SVTK$, 4-61
UMAP$, 4-6
USTP$, 4-62
WSIG$S, 4-64
WTLO$, 4-65
WTSE$, 4-66
Magnetic tape,
ACP, 5-23
driver, 5-9
Mailbox,
create, 3-3, 3-10 to 3-11
I/O, 3-10
information, 4-26
name, 3-10
read from, 3-10 to 3-11, 4-38, 4-40 to 4-41,

Mailbox (Cont.)
4-54 to 4-55
receive from, 4-54
send to, 3-10 to 3-11, 4-48
MAP ADDRESS WINDOW directive, 4-4
MAP$ macro, 4-4
Mapping,
device name between VAX/VMS and RSX-11M, 3-7 to 3-8
pseudodevice name, 3-8
Mark time,
cancel, 4-13
MARK TIME directive, 2-8, 4-33
MCR command line, 4-28
Member,
UIC, 2-1
Memory management, 2-6
Mount ($MOUNT) system service, 3-4
MRKT$ macro, 4-33
Multiuser task,
image, 1-4
name, 2-3

Name,
device, 3-7 to 3-8, 4-10
logical device, 3-7
multiuser task image, 2-3
partition, 2-7, 4-32
physical device, 4-10
process, 2-3 to 2-4, 4-32
PRT... for task image, 3-12
task image, 2-3 to 2-4, 3-10
Nonshareable device, 3-4, 5-2
Null device, 5-23

OV pseudodevice, 3-8
Overlay, 1-4

Pager, 2-7
Parent/offspring task directive,
EXIT WITH STATUS, 4-24
RECEIVE DATA OR STOP, 4-38
SPAWN, 4-54
STOP, 4-58
STOP FOR LOGICAL OR OF EVENT FLAGS, 4-57
STOP FOR SINGLE EVENT FLAG, 4-59
UNSTOP TASK, 4-62
Parse,
file specification, 2-13, B-1
Partition,
information, 4-30
name, 2-7, 4-30, 4-32
support under VAX/VMS, 2-7
Physical device,
conversion, 3-7
information, 4-26
mapping, 3-7
name, 4-10
queue, 4-35, 4-37
supported, 5-2
PLAS directive, 1-4
Power recovery AST, 4-53
PRINT$ macro, 3-12
Priority,
process, 2-8, 4-10
software, 2-8
swapper, 2-6
task, 4-32
Privilege, 2-1
Process,
detached, 2-10
identification, 2-3
information, 4-32
name, 2-3 to 2-4, 4-32
priority, 4-9
protection, 2-1
subprocess, 2-10, 4-54
UIC, 4-32
VAX/VMS, 1-5
virtual address space, 1-5 to 1-6
Processor mode, 1-1
Program region, 1-5
Programming environment, 1-5
Protection,
event flag, 2-4
mailbox, 3-10
process, 2-1
PRT... task name, 3-12
Pseudodevice,
CL, 3-8
CO, 3-8
LB, 3-8 to 3-9
OV, 3-8
SP, 3-8
SY, 3-8
TI, 3-8
WK, 3-8

QIO$ macro, 4-35
QIOW$ macro, 4-37

QUEUE I/O REQUEST AND WAIT directive, 4-37
QUEUE I/O REQUEST directive, 3-12, 4-35, 5-1, 5-3
Queue I/O Request ($QIO) system service, 3-3, 4-35, 4-38, 4-40 to 4-41, 4-48, 5-1
Queue I/O Request & Wait for Event Flag ($QIOW) system service, 4-37

RCST$ macro, 4-38
RCVD$ macro, 4-40
RCVX$ macro, 4-41
RDAF$ macro, 4-42
RDXF$ macro, 4-43
READ ALL EVENT FLAGS directive, 4-42
Read Event Flags ($READEF) system service, 4-22, 4-42 to 4-43
READ EXTENDED EVENT FLAGS directive, 4-43
RECEIVE BY DIFFERENCE directive, 4-5
Receive data AST, 4-55
RECEIVE DATA directive, 3-10, 4-40
RECEIVE DATA OR EXIT directive, 3-10, 4-41
RECEIVE DATA OR STOP directive, 3-10, 4-38
Record Management Services for the PDP-11 (RMS-11), See RMS-11
Region,
  control, 1-5
  permanent, 1-4
  program, 1-5
  shareable, 1-4
  temporary, 1-4
REQUEST TASK directive, 2-10, 4-44
RESCOM option, 2-9
RESET instruction, 1-2
RESLIB option, 2-9
Resource usage limits, 2-2
RESUME TASK directive, 2-10, 4-45
RMS-11, 3-5
RQST$ macro, 4-44
RREF$ macro, 4-5
RSUM$ macro, 4-45
RSX-11M Directive, See Directive
RSX-11M Macro, See Macro
RSXCOMEFN event flag cluster, 2-4
$RSXDEF macro, 2-11
RSXGROUPEFN event flag cluster, 2-5
RUN command, 2-2, 4-29
RUN TASK directive, 2-10, 4-46
RUN$ macro, 4-46

Schedule Wake-up ($SCHDWK) system service, 4-46
SDAT$ macro, 4-48
Section,
  global, 2-8
SEND BY REFERENCE directive, 4-5
SEND DATA directive, 3-10, 3-12, 4-48
Send Message to Symbiont Manager ($SNDSMB) system service, 3-12
SET EVENT FLAG directive, 4-50
Set Event Flag ($SETEF) system service, 4-50
Set Power Recovery AST ($SETPRA) system service, 4-53
Set Process Swap Mode ($SETSWM) system service, 4-21
Set Process Swap Mode($SETSWM) system service, 4-18
SET TERMINAL command, 5-16
Set Timer ($SETIMR) system service, 4-33
SETF$ macro, 4-50
SFPA$ macro, 4-51
Shareable device, 3-4, 5-2
Shareable region, 1-4
Significant event, 4-16
  declare, 4-16
  support under VAX/VMS, 2-7
  wait for, 4-64
SP pseudodevice, 3-8
SPAWN directive, 4-54
SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST directive, 4-51
SPECIFY POWER RECOVERY AST directive, 4-53
SPECIFY RECEIVE DATA AST directive, 4-55
SPECIFY RECEIVE-BY-REFERENCE AST directive, 4-5

SPECIFY SST VECTOR TABLE FOR DEBUGGING AID directive, 4-60
SPECIFY SST VECTOR TABLE FOR TASK directive, 4-61
SPND$S macro, 4-52
Spool function,
  FCS, 3-12
Spooled device, 3-11 to 3-12, 5-11, 5-22
SPRA$ macro, 4-53
SPWN$ macro, 4-54
SRDA$ macro, 4-55
SREF$ macro, 4-5
SRRA$ macro, 4-5
SST (Synchronous System Trap),
  service routine, 2-11
  specify vector table, 4-60 to 4-61
  vector table, 2-11
  vector table information, 4-32
Status,
  I/O, 4-35, 4-37
  I/O block, 5-3
STLO$ macro, 4-57
STOP directive, 4-58
STOP FOR LOGICAL OR OF EVENT FLAGS directive, 4-57
STOP FOR SINGLE EVENT FLAG directive, 4-59
STOP$S macro, 4-58
STSE$ macro, 4-59
Subprocess, 2-10, 4-54
SUSPEND directive, 4-52
SVDB$ macro, 4-60
SVTK$ macro, 4-61
Swapper,
  disable, 4-18
  enable, 4-21
  function, 2-6 to 2-7
SY pseudodevice, 3-8
Synchronous System Trap (SST),
  See SST (Synchronous System Trap)
SYS$COMMAND device, 3-8
SYS$DISK device, 3-8
SYS$ERROR device, 2-11, 3-8
SYS$INPUT device, 3-8
SYS$OUTPUT device, 2-11, 3-8
SYS$SCRATCH device, 3-8
SYS$SYSROOT device, 3-8 to 3-9
System,
  clock, 2-8, 4-31
  status code, 4-7, 4-24
System environment, 1-5, 2-1
System service,
  Allocate Device ($ALLOC), 3-4, 5-2
System service (Cont.)
  Assign I/O Channel ($ASSIGN), 3-3, 4-10
  Associate Common Event Flag Cluster ($ASCEFC), 2-4, 4-14
  Cancel I/O on Channel ($CANCEL), 3-4, 5-3, 5-23
  Cancel Timer Request ($CANTIM), 4-13
  Cancel Wakeup ($CANWAK), 4-15
  Clear Event Flag ($CLREF), 4-12
  Create Mailbox and Assign I/O Channel ($CREMBX), 3-3
  Disassociate Common Event Flag Cluster ($DACEFC), 4-19
  Dismount ($DISMOU), 3-4
  Exit ($EXIT), 2-11, 4-22 to 4-24, 4-41
  for I/O, 3-2
  Force Exit ($FORCEX), 4-8
  Get Device/Volume Information ($GETDVI), 3-4
  Get I/O Channel Information ($GETCHN), 4-27
  Get Time ($GETTIM), 4-31
  Hibernate ($HIBER), 4-38 to 4-39, 4-52, 4-58
  Mount ($MOUNT), 3-4
  Queue I/O Request and Wait for Event Flag ($QIOW), 4-37
  Queue I/O Request ($QIO), 3-3, 4-35, 4-38, 4-40 to 4-41, 4-48, 5-1
  Read Event Flags ($READEF), 4-22, 4-42 to 4-43
  Schedule Wake-up ($SCHDWK), 4-46
  Send Message to Symbiont Manager ($SNDSMB), 3-12
  Set Event Flag ($SETEF), 4-50
  Set Power Recovery AST ($SETPRA), 4-53
  Set Process Swap Mode ($SETSWM), 4-18, 4-21
  Set Timer ($SETIMR), 4-33
  Wait for Logical OR of Event Flags ($WFLOR), 4-57, 4-59, 4-65 to 4-66
  Wake ($WAKE), 4-44 to 4-45, 4-62

Task Builder, 1-2 to 1-3, 2-9
Task execution control directive,
  ABORT TASK, 4-8
  CANCEL TIME BASED INITIATION REQUESTS, 4-15
  EXTEND TASK, 4-25
  REQUEST TASK, 4-44
  RESUME TASK, 4-45
  RUN TASK, 4-46
  SUSPEND, 4-52
  TASK EXIT, 4-23
TASK EXIT directive, 4-23
Task image,
  extend, 4-25
  hibernate, 2-10, 4-38, 4-44 to 4-46, 4-58
  installed, 2-3
  interface to I/O system, 3-6
  multiuser, 1-4
  name, 2-3
  parameter, 4-32
  priority, 4-32
  size, 4-25
  SST vector table, 4-61
  stop, 2-10
  suspend, 2-10, 4-52
  termination, 2-10 to 2-11, 4-8, 4-22 to 4-24, 4-41
  wake, 2-10, 4-45 to 4-46, 4-62
Task status control directive,
  ALTER PRIORITY, 4-9
  DISABLE CHECKPOINTING, 4-18
  ENABLE CHECKPOINTING, 4-21
Terminal driver,
  differences in <CR> and <LF> use under VAX/VMS, 5-15
  function code correspondence, 5-12
  IO.ATT function, 5-15
  IO.CCO function, 5-18
  IO.DET function, 5-16
  IO.GTS function, 5-19 to 5-20
  IO.KIL function, 5-16
  IO.RAL function, 5-16
  IO.RLB function, 5-16
  IO.RNE function, 5-16
  IO.RPB function, 5-17 to 5-18
  IO.RPR function, 5-18
  IO.RST function, 5-16
  IO.RTT function, 5-16
  IO.RVB function, 5-18
  IO.WBT function, 5-18
  IO.WLB function, 5-18
  IO.WPB function, 5-19
  IO.WVB function, 5-19
Terminal driver (Cont.)
  parameter correspondence, 5-13
  read status return information, 5-22
  SF.GMC function, 5-21
  SF.SMC function, 5-21
  subfunction bit correspondence, 5-14
  terminal characteristics, 5-21
  with process-permanent file, 5-11
Termination,
  abnormal, 2-11
  AST, 4-54
  AST service routine, 4-11
  code, 2-11
  image, 2-10 to 2-11, 4-8, 4-22 to 4-24, 4-41
  normal, 2-10
Termination code, 2-10
TI pseudodevice, 3-8
Time,
  delay, 4-33, 4-46
  information, 4-31
Time-synchronized wake request cancellation, 4-15
Trap-associated directive,
  AST SERVICE EXIT, 4-11
  DISABLE AST RECOGNITION, 4-17
  ENABLE AST RECOGNITION, 4-20
  INHIBIT AST RECOGNITION, 4-17
  SPECIFY FLOATING-POINT PROCESSOR EXCEPTION AST, 4-51
  SPECIFY POWER RECOVERY AST, 4-53
  SPECIFY RECEIVE DATA AST, 4-55
  SPECIFY SST VECTOR TABLE FOR DEBUGGING AID, 4-60
  SPECIFY SST VECTOR TABLE FOR TASK, 4-61

UIC (User Identification Code),
  group, 2-1
  information, 2-2
  member, 2-1
  process, 2-1, 4-32
UMAP$ macro, 4-6
Unit,
  logical, 4-10, 4-26, 4-32
Unknown terminal device, 5-21

UNMAP ADDRESS WINDOW
directive, 4-6
UNSTOP TASK directive, 2-10,
4-62
Unsupported directive,
added since RSX-11M Version
3.2, 4-1
ATTACH REGION, 4-2
CONNECT, 4-2
CONNECT TO INTERRUPT VECTOR,
4-2
CREATE ADDRESS WINDOW, 4-2
CREATE REGION, 4-2
DETACH REGION, 4-2
ELIMINATE ADDRESS WINDOW,
4-3
GET MAPPING CONTEXT, 4-3
GET REGION PARAMETERS, 4-3
GET SENSE SWITCHES, 4-3
MAP ADDRESS WINDOW, 4-4
RECEIVE BY REFERENCE, 4-5
SEND BY REFERENCE, 4-5
SPECIFY RECEIVE-BY-REFERENCE
AST, 4-5
UNMAP ADDRESS WINDOW, 4-6
User authorization file (UAF),
2-1 to 2-2, 4-54
User Identification Code
(UIC),
See UIC (User Identification
Code)
USTP$ macro, 4-62
VAX-11 RMS (Record Management
Services), 3-1
VAX/VMS system services,
See System services
Version number,
negative, 2-13
Virtual address location,
for RSX-11M task image, 1-5
Virtual address space, 1-5,
2-7
WAIT FOR LOGICAL OR OF EVENT
FLAGS directive, 4-65
Wait for Logical OR of Event
Flags ($WFLOR) system
service, 4-57, 4-59,
4-65 to 4-66
WAIT FOR SIGNIFICANT EVENT
directive, 2-7, 4-64
WAIT FOR SINGLE EVENT FLAG
directive, 4-66
Wake ($WAKE) system service,
4-44 to 4-45, 4-62
WK pseudodevice, 3-8
Working set, 2-6
WSIG$S macro, 4-64
WTLO$ macro, 4-65
WTSE$ macro, 4-66

VAX-11/RSX-11M
Programmer's Reference Manual
AA-D020C-TE

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

Do Not Tear - Fold Here

Cut Along Dotted Line